

Queueing Theory for LLM Serving

Foundations, Agentic Sessions, and Prefill–Decode Disaggregation

Serving Queue Theory Project

September 2026

Contents

About These Notes	3
1 Probabilistic Foundations: Arrivals, Service, and Little's Law	5
1.1 Queueing systems and performance measures	5
1.2 The colocated replica as queues	6
1.3 The Poisson process	8
1.4 PASTA: Poisson arrivals see time averages	9
1.5 The renewal-reward theorem	11
1.6 Little's law	11
1.7 Summary	12
1.8 Exercises	13
2 Markovian Queues: M/M/1, Processor Sharing, and Insensitivity	14
2.1 Continuous-time Markov chains	14
2.2 Birth–death processes and M/M/1	15
2.3 Multi-server queues and memory slots	16
2.4 Processor sharing and state-dependent capacity	17
2.5 Insensitivity	18
2.6 The price at the decode stage	20
2.7 Summary	20
2.8 Exercises	20
3 The M/G/1 Queue and the Pollaczek–Khinchine Formula: the Price of a Miss	22
3.1 The M/G/1 queue and the residual service time	22
3.2 The Pollaczek–Khinchine formula	23
3.3 Mixtures of hits and misses	24
3.4 The price of a miss	25
3.5 Summary	27
3.6 Exercises	27
4 Closed Systems and Memory: Finite Populations, MVA, Campbell's Theorem, Eviction	28
4.1 Closed systems and mean value analysis	28
4.2 M/G/ ∞ and Campbell's theorem	30
4.3 Eviction: the covering knapsack problem	32
4.4 Summary	33
4.5 Exercises	34

5	Colocated Serving: The Paper's Scheduler, Step by Step	35
5.1	The model on one page	35
5.2	Where the variance of prefill work comes from	36
5.3	The decision problem and one price	36
5.4	The scheduler, step by step	39
5.5	What is exact and what is approximate	40
5.6	Summary	41
5.7	Exercises	41
6	Disaggregated Serving	43
6.1	The disaggregated replica	43
6.2	Sample paths	44
6.3	Three prices	45
6.4	The decision problem with two pools	46
6.5	How long to keep	46
6.6	How many sessions to admit	47
6.7	Misses that feed themselves	47
6.8	What is exact and what is approximate	48
6.9	Summary	49
6.10	Exercises	49
7	Metastability of a Prefix Cache Under Load	50
7.1	The replica	50
7.2	What LRU keeps under FCFS	51
7.3	The closed replica: the congested queue and an admission rule	51
7.4	Admit, then hold	53
7.5	What a predictive model must keep	53
7.6	Experiments	55
7.7	Conclusions	57
7.8	What is exact and what is approximate	59
7.9	Exercises	60
A	Executable Examples in serQ	61
A.1	What to run and what it checks	61

About These Notes

These notes are a course on queueing theory for serving large language models. They assume undergraduate probability and real analysis (conditional expectation, the strong law of large numbers, power series), but no prior queueing theory. Definitions, proofs and exercises for the common tools appear once. Chapters 5 and 6 apply them to a colocated replica and a prefill–decode deployment, and Chapter 7 to metastable failures.

The system. Prefill reads a prompt and produces the first token. Decode produces the remaining tokens. Both need KV memory. A session returns with another turn after a tool call, so session arrivals and turn arrivals are different processes. A cache policy changes how much prefill work a turn needs. A colocated engine shares iterations between prefill and decode; a disaggregated engine transfers KV between their separate memory pools.

Reading plan.

- Chapter 1: the colocated replica as queues, Poisson arrivals, PASTA, renewal rewards and Little’s law.
- Chapter 2: Markov chains, M/M/1, memory slots, processor sharing and insensitivity.
- Chapter 3: the Pollaczek–Khinchine formula and the price of a miss.
- Chapter 4: finite populations, resident memory and eviction.
- Chapter 5: the colocated scheduler and its holding-time rule.
- Chapter 6: the disaggregated replica, transfer prices, two memory pools and cache feedback.
- Chapter 7: metastability of the cache feedback, recovery after a burst, and how interventions rank.
- Appendix A: how to run the executable examples, and what they check.

Chapters 1 to 4 are the common tools. A reader interested only in colocated serving can stop after Chapter 5.

Models and programs. The mathematical queues are idealisations with explicit assumptions. A step engine with block admission and preemption does not inherit their formulas automatically. The executable examples in Appendix A make some of these engine rules explicit. The paper these notes follow concerns colocated serving; Chapter 6 extends its theory to disaggregated serving. The exercises reinforce the proofs; [13, 6] are introductory references and [2] gives a rigorous treatment.

Notation. Symbols are defined where they first appear. The table lists those whose meaning changes between chapters.

Symbol	Meaning
W_q, W	wait before service; in Chapters 1 and 2 W is the sojourn time, from Chapter 3 on the PK wait
D	decode work: seconds in Chapter 5, tokens in Chapter 6
$\mathcal{B}$	byte-second budget (Chapters 5 and 6)
γ	μZ in Chapter 4
B, X	link bandwidth; transfer work (Chapter 6)
h, H	hit rate; hit indicator, or the feedback map in Section 6.7

Chapter 1

Probabilistic Foundations: Arrivals, Service, and Little's Law

This chapter defines the vocabulary of queueing theory and proves three tools that every later chapter relies on: the memorylessness of the exponential distribution, the Poisson process together with PASTA, and Little's law.

1.1 Queueing systems and performance measures

A **queueing system** has four elements. **Customers** (or jobs) arrive, require some amount of processing, and leave; the arrival time of the k -th customer is a_k , numbered so that $0 \leq a_1 \leq a_2 \leq \dots$. The **arrival process** is the sequence of arrival times $(a_k)_{k \geq 1}$, or equivalently the counting process $A(t) = \#\{k : a_k \leq t\}$. A **server** is the resource that processes customers: customer k needs time S_k when it has the server to itself, its **service time**, also called its **work** or service requirement. Customers who arrive at a busy server wait in a **waiting room**, and a **service discipline** decides whom to serve, when, and how much.

Two disciplines recur in these notes. **FIFO** (first in, first out) serves one customer at a time, to completion, in order of arrival. **Processor sharing** (PS) serves all customers present at once at equal rates (Definition 2.9). A discipline under which the server never idles while work is present is **work-conserving**; both FIFO and PS are. A **delay station** (infinite-server station) has as many servers as customers: an arriving customer starts service at once, so its sojourn time is its own service time, whatever the other customers do. A **closed system** is one in which a fixed number N of customers circulate among stations forever, with no entry from or exit to the outside; the time a customer of a closed system spends at a delay station is its **think time**, with mean Z .

Definition 1.1 (Performance measures)

Let d_k be the time at which customer k leaves the system.

1. The **sojourn time** (response time) is $W_k = d_k - a_k$.
2. The **waiting time** $W_{q,k}$ is the time from the arrival of customer k until its service first begins. For a FIFO single server, $W_k = W_{q,k} + S_k$.
3. The **number in system** is $N(t) = \#\{k : a_k \leq t < d_k\}$. The number in the waiting room is written $N_q(t)$.

Quantities in queueing theory are defined by two different kinds of averages, and the distinction will be essential several times.

Definition 1.2 (Time averages and customer averages)

The **time average** of the process $N(t)$ is $L = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t N(s) ds$, and the **customer average** of the sequence (W_k) is $W = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n W_k$ (when the limits exist). L_q and W_q are defined in the same way.

When the system is stationary and ergodic, the limits are almost surely constant and equal $\mathbb{E}[N]$ and $\mathbb{E}[W]$ under the stationary law; this is how a time average computed from a model is compared with a mean measured over a long run.

Definition 1.3 (Arrival rate, load, stability)

The **arrival rate** is $\lambda = \lim_{t \rightarrow \infty} A(t)/t$. The **load** (utilisation) of a single-server system with mean service time $\mathbb{E}[S]$ is

$$\rho = \lambda \mathbb{E}[S].$$

ρ is the amount of work arriving per unit time and is dimensionless. The system is **stable** if $N(t)$ converges in distribution, as $t \rightarrow \infty$, to an almost surely finite random variable.

A single server processes at most one unit of work per unit time, so if $\rho > 1$ the unfinished work grows linearly and the system cannot be stable. For a work-conserving G/G/1 system (i.i.d. interarrival and service times) it is known that $\rho < 1$ is essentially necessary and sufficient for stability [16]. For the models in these notes we assume $\rho < 1$ as the stability condition and verify it model by model.

Kendall notation. A queueing system is written $A/S/c/K/N$: A is the law of the interarrival times, S the law of the service times, c the number of servers, K the system capacity and N the size of the customer population, the last two ∞ when omitted. The symbols for the laws are M (Markov: exponential and independent), D (deterministic: constant) and G (general: an arbitrary law, i.i.d.). The queues used in these notes are collected here once; each lecture refers back to this list.

Definition 1.4 (The queues of these notes)

In (a)–(e) arrivals form a Poisson process of rate λ (Definition 1.8), the service times are i.i.d., independent of the arrivals, with generic S and $\mathbb{E}[S] < \infty$, and the waiting room is infinite.

- (a) **M/M/1**: one FIFO server and $S \sim \text{Exp}(\mu)$ (Definition 1.5); its load is $\rho = \lambda/\mu$ (Chapter 2).
- (b) **M/M/m**: m identical servers, $S \sim \text{Exp}(\mu)$, customers served in arrival order; a customer waits only when all m servers are busy. Its **offered load** $a = \lambda/\mu$ is the mean number of busy servers, and the load per server is a/m (Section 2.3).
- (c) **M/G/1**: one FIFO server and an arbitrary law G of S ; when the server becomes free it serves the earliest waiting customer to completion without interruption (Chapter 3).
- (d) **M/G/1-PS**: one server with an arbitrary law G of S , under processor sharing (Definition 2.9).
- (e) **M/G/ ∞** : a delay station with an arbitrary law G of S (Section 4.2).
- (f) **M/M/1//N**, the **finite-source** or machine-repairman model: a closed system of N customers and two stations, a delay station with exponential think times of mean Z and a FIFO server with service times $\text{Exp}(\mu)$. Each customer goes to the server when its think time ends and starts thinking again when its service ends; all think and service times are mutually independent (Section 4.1).

1.2 The colocated replica as queues

A queueing model of a serving system keeps what decides which request waits and for how long, and leaves out the rest. Figure 1.1 shows the model of a colocated replica used throughout these notes; its stations are defined formally in later chapters.

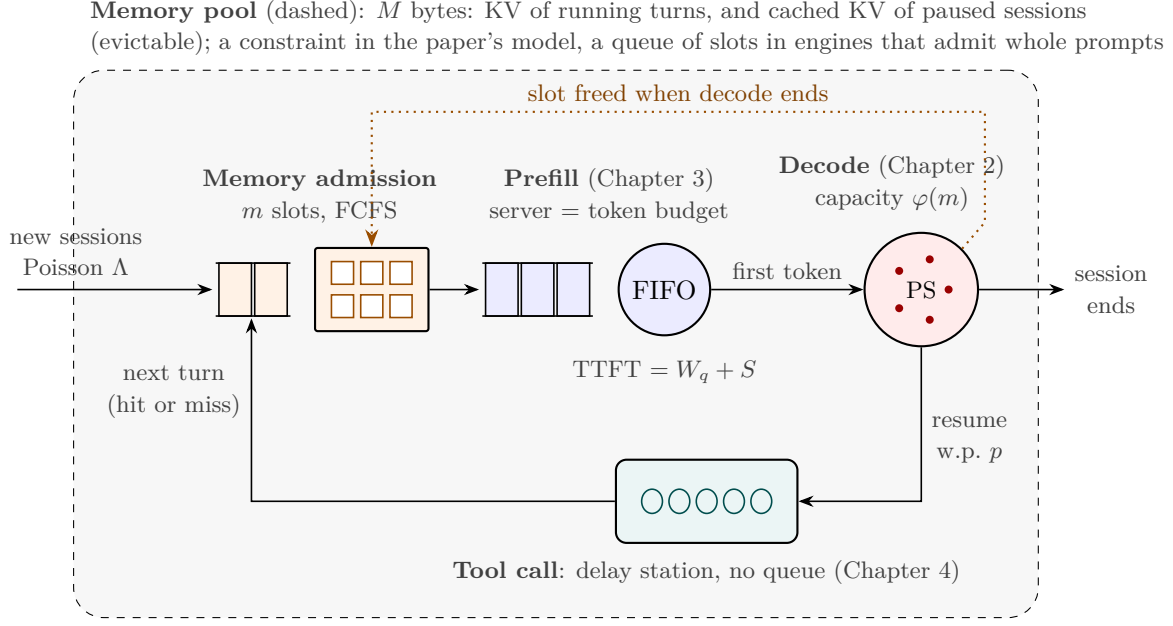


Figure 1.1: A colocated replica as a queueing model. Each turn (one request to the engine) passes through prefill and decode; each session loops through the tool call until the agent program ends. The dashed pool is shared memory: what it drops decides whether the next turn is a hit (short prefill work) or a miss (long prefill work). The memory-admission station (orange) is *not* in the paper's model: it appears in engines that admit a turn only when its whole context fits, in arrival order; a turn then holds a slot from the start of its prefill to the end of its decode (Section 5.5).

Requests and sessions. A *turn* is one request to the engine: the agent sends its context, the engine prefills it and decodes the reply, and the request completes. A *session* is one run of the agent program: it sends a turn, waits for the reply, calls a tool, sends the next turn with the tool output appended, and ends when the program ends. In the vocabulary of Section 1.1 the turns are the customers of the prefill and decode stations, and the sessions are the customers of the loop through the tool call. New sessions arrive as a Poisson process of rate Λ , but a session's next turn exists only after its previous one has been served, so turn arrivals depend on the state.

Stations and memory. *Prefill* is a FIFO server whose capacity is the token budget of an engine step, given to the oldest pending prefill first: the M/G/1 queue of Chapter 3, or the closed system of Chapter 4 with N live sessions. *Decode* is a processor-sharing station: each step advances every turn in the batch by one token, at a total rate $\varphi(m)$ for batch size m (Chapter 2). The *tool call* is a delay station. In the paper's model the memory pool of M bytes is only a constraint on resident state (Chapter 4). Engines such as vLLM, which admit a turn only when its whole context fits and in arrival order, make memory a queue of slots as well (Section 2.3); on the paper's long-context testbed this queue set the waits (Section 5.5).

The cache chooses the service law. In classical queueing theory the service-time law is given. Here the KV policy chooses it: a hit has short prefill work, a miss recomputes the whole context. The TTFT is the sojourn time at prefill, $W_q + S$, and the objective is $L = L_P + L_D$, which by Little's law (Theorem 1.17) is the mean delay per unit time. The model measures work in seconds, averages the budget the decode batch leaves to prefill, and treats turn arrivals as Poisson; Chapter 5 returns to these gaps, and Chapter 6 draws the same picture for disaggregated serving.

1.3 The Poisson process

Definition 1.5 (Exponential distribution)

A random variable $X \geq 0$ with $\mathbb{P}(X > t) = e^{-\mu t}$ for $t \geq 0$ has the **exponential distribution** with **rate** $\mu > 0$, written $X \sim \text{Exp}(\mu)$.

Integrating the tail gives $\mathbb{E}[X] = \int_0^\infty e^{-\mu t} dt = 1/\mu$ and $\mathbb{E}[X^2] = \int_0^\infty 2t e^{-\mu t} dt = 2/\mu^2$.

Proposition 1.6 (Memorylessness [22])

If $X \sim \text{Exp}(\mu)$, then for all $s, t \geq 0$

$$\mathbb{P}(X > s + t \mid X > s) = \mathbb{P}(X > t). \quad (1.1)$$

Conversely, if a random variable with $0 < X < \infty$ a.s. satisfies (1.1), then $X \sim \text{Exp}(\mu)$ for some $\mu \in (0, \infty)$.

Proof. The forward direction is $e^{-\mu(s+t)}/e^{-\mu s} = e^{-\mu t}$. Converse: let $G(t) = \mathbb{P}(X > t)$. Then (1.1) reads $G(s+t) = G(s)G(t)$. G is nonincreasing and right-continuous, and $G(0) = \mathbb{P}(X > 0) = 1$. By induction $G(nt) = G(t)^n$, hence $G(1) = G(1/n)^n$ and $G(m/n) = G(1/n)^m$. If $G(1) = 0$, then $G(1/n) = 0$ for every n , which contradicts $G(1/n) \rightarrow G(0) = 1$ by right-continuity. If $G(1) = 1$, then $G(r) = 1$ for every rational r , and by monotonicity $G \equiv 1$, that is $X = \infty$ a.s., contradicting the assumption. Hence $G(1) = e^{-\mu}$ for some $\mu \in (0, \infty)$, and $G(r) = e^{-\mu r}$ on the rationals. For arbitrary t , take rationals $r \downarrow t$; right-continuity gives $G(t) = e^{-\mu t}$. $\square$

Memorylessness says that the remaining service time of a customer who has already received s units of service has the same distribution as the service time of a new customer. This is why, in a system with exponential service, the number in system $N(t)$ alone determines the future, and $N(t)$ is a continuous-time Markov chain (Chapter 2). The following fact is also used often.

Lemma 1.7 (Minimum of exponentials [22])

If $X_i \sim \text{Exp}(\mu_i)$, $i = 1, \dots, n$, are independent, then $\min_i X_i \sim \text{Exp}(\sum_i \mu_i)$.

Proof. $\mathbb{P}(\min_i X_i > t) = \prod_i \mathbb{P}(X_i > t) = e^{-(\sum_i \mu_i)t}$. $\square$

Memorylessness at the server gives Markovian service; at the arrivals it gives the Poisson process: a stream whose interarrival times are independent and exponential (Theorem 1.9).

Definition 1.8 (Counting process and Poisson process)

A **counting process** $A = (A(t))_{t \geq 0}$ is a nondecreasing, right-continuous, $\mathbb{N}$ -valued process with $A(0) = 0$ and jumps of size one only. A counting process A is a **Poisson process** with **rate** $\lambda > 0$ if

1. (**independent increments**) for disjoint intervals $(s_1, t_1], \dots, (s_n, t_n]$ the increments $A(t_j) - A(s_j)$ are independent;
2. (**stationary Poisson increments**) for all $s \geq 0, t > 0$, $A(s+t) - A(s) \sim \text{Poisson}(\lambda t)$, that is $\mathbb{P}(A(s+t) - A(s) = n) = e^{-\lambda t} (\lambda t)^n / n!$.

$S_n = \inf\{t : A(t) \geq n\}$ is the n -th arrival time, and $T_n = S_n - S_{n-1}$ ($S_0 = 0$) is the n -th **interarrival time**.

Theorem 1.9 (Exponential interarrival times [22, §2.2])

A is a Poisson process with rate λ if and only if the interarrival times $T_1, T_2, \dots$ are independent and each is $\text{Exp}(\lambda)$.

Proof ($\Rightarrow$). For $0 = t_0 < t_1 < \dots < t_n$ and small $h > 0$ consider the event $E_h = \{S_j \in (t_j, t_j + h], j = 1, \dots, n\}$. Let F_h be the event that there is no arrival in $(0, t_1]$ or in $(t_{j-1} + h, t_j]$ for $j \geq 2$, and exactly one in each $(t_j, t_j + h]$. Then $F_h \subset E_h$, and $E_h \setminus F_h$ is contained in the event that some interval of length h has two or more arrivals, which has probability $O(h^{n+1})$. By independent and Poisson increments,

$$\mathbb{P}(F_h) = e^{-\lambda(t_1 + (t_2 - t_1 - h) + \dots + (t_n - t_{n-1} - h))} (\lambda h e^{-\lambda h})^n = \lambda^n h^n e^{-\lambda t_n} (1 + O(h)).$$

Hence $(S_1, \dots, S_n)$ has density $\lambda^n e^{-\lambda s_n}$ on $\{0 < s_1 < \dots < s_n\}$. The linear map $T_j = S_j - S_{j-1}$ has Jacobian one and $s_n = \sum_j T_j$, so $(T_1, \dots, T_n)$ has density $\prod_{j=1}^n \lambda e^{-\lambda t_j}$, that is, independent $\text{Exp}(\lambda)$. The converse is a standard argument using that S_n is Gamma distributed together with memorylessness, and we omit it [22, §2.2]. $\square$

Proposition 1.10 (Superposition and thinning [11])

- (i) (**Superposition**) If A_1, A_2 are independent Poisson processes with rates λ_1, λ_2 , then $A_1 + A_2$ is a Poisson process with rate $\lambda_1 + \lambda_2$.
- (ii) (**Thinning**) If each point of a Poisson process with rate λ is kept with probability p and discarded with probability $1 - p$, independently of everything else, then the kept points and the discarded points are Poisson processes with rates $p\lambda$ and $(1-p)\lambda$, and they are independent.

Proof. See Exercise 1.1. $\square$

In LLM serving

If sessions arrive as a Poisson process of rate Λ and each turn is routed independently to one of J replicas, then by Proposition 1.10(ii) each replica also sees Poisson arrivals. The turns within a session, however, go around a closed loop (prefill $\rightarrow$ decode $\rightarrow$ tool), so the turn arrivals are not exactly Poisson. The closed models of Chapter 4 are the tool for this gap.

1.4 PASTA: Poisson arrivals see time averages

The distribution of the state seen by arriving customers generally differs from the time-average distribution (Example 1.12(b) below gives an example). With Poisson arrivals this discrepancy disappears.

Definition 1.11 (Lack of anticipation)

Given a state process $X = (X(t))_{t \geq 0}$ and an arrival process A , we say that A has **lack of anticipation** with respect to X if for every t the future increments $\{A(t+s) - A(t) : s \geq 0\}$ are independent of the past $\{X(u) : u \leq t\} \cup \{A(u) : u \leq t\}$.

In a queueing system this assumption says that future arrivals are independent of the system state so far. Note the direction: the state may (and does) depend on past arrivals; what is excluded is that future arrivals react to the state. Poisson arrivals from outside satisfy it naturally. Arrivals in a closed system (Chapter 4) do not.

Example 1.12 (One process with lack of anticipation, one without)

(a) *Lack of anticipation: the M/M/1 queue.* Customers come from a large outside population as a Poisson process, independently of the service times. The queue length $X(t)$ is a function of the arrivals before t and the service times, and by independent increments the arrivals after t are independent of both. An arriving customer is like a raindrop falling on the queue: the rain does not know how long the queue is. So the arrival instants are “random inspection times”, and by Theorem 1.14 the fraction of arrivals that find the server busy equals the fraction of time it is busy, ρ .

(b) *No lack of anticipation: a single agent session.* One session alternates between a tool call (mean Z) and a turn at a server (mean $1/\mu$). Its next turn can arrive only after its previous turn has left the server. If $X(t) = 1$ (the turn is being served), then no arrival can happen until the service ends; if $X(t) = 0$, an arrival may come at any moment. Future arrivals depend on the current state, so the condition fails. The consequence is visible: every arrival finds the server idle, while the server is busy a fraction $(1/\mu)/(Z + 1/\mu) > 0$ of the time. Arrivals do not see time averages. With N sessions the same effect is weaker but present, and Chapter 4 replaces PASTA by the arrival theorem for exactly this reason.

Remark 1.13 (Why study PASTA if sessions violate it?)

Three reasons. The M/G/1 model of prefill and its Pollaczek–Khinchine formula (Theorem 3.4) rest on PASTA, so one must know what session feedback approximates. The superposition of many independent sessions is close to Poisson (the Palm–Khinchine theorem [10]), and new sessions are Poisson in the model. And the exact closed-system answer, the arrival theorem (Theorem 4.1), is a corrected PASTA that reduces to it as the number of sessions grows.

Theorem 1.14 (PASTA [26])

Let A be a Poisson process with rate λ , X a state process with left limits, and B a set of states, and suppose A has lack of anticipation with respect to X . If $U(t) = \mathbf{1}\{X(t^-) \in B\}$ is left-continuous, then for every $T > 0$

$$\mathbb{E}\left[\sum_{k: S_k \leq T} \mathbf{1}\{X(S_k^-) \in B\}\right] = \lambda \mathbb{E}\left[\int_0^T \mathbf{1}\{X(t) \in B\} dt\right]. \quad (1.2)$$

Consequently, if the limits of the two averages $\frac{1}{T} \int_0^T \mathbf{1}\{X(t) \in B\} dt$ and $\frac{1}{A(T)} \sum_{k \leq A(T)} \mathbf{1}\{X(S_k^-) \in B\}$ exist with probability one, they are equal: the fraction of arrivals that find the system in B equals the fraction of time it spends in B .

Proof. The left side is $\int_0^T U(t) dA(t)$. For the partition $t_j = jT/n$ of $[0, T]$, define the step function $U_n(t) = \sum_j U(t_j) \mathbf{1}\{t \in (t_j, t_{j+1}]\}$. $U(t_j)$ is determined by the past up to time t_j , and by lack of anticipation the increment $A(t_{j+1}) - A(t_j)$ is independent of that past, so

$$\mathbb{E}\left[\int_0^T U_n dA\right] = \sum_j \mathbb{E}[U(t_j)] \mathbb{E}[A(t_{j+1}) - A(t_j)] = \lambda \sum_j \mathbb{E}[U(t_j)] (t_{j+1} - t_j) = \lambda \mathbb{E}\left[\int_0^T U_n(t) dt\right].$$

Since U is left-continuous, $U_n(t) \rightarrow U(t)$ for every t . The integrands on the two sides are bounded by $A(T)$ and T respectively, and $\mathbb{E}[A(T)] = \lambda T < \infty$, so by dominated convergence $\mathbb{E}[\int_0^T U_n dA] = \lambda \mathbb{E}[\int_0^T U dt]$ as $n \rightarrow \infty$. Finally, the set of t with $X(t^-) \neq X(t)$ is countable, so replacing $U(t)$ by $\mathbf{1}\{X(t) \in B\}$ on the right does not change the integral. The statement about limits follows from a sample-path version of (1.2) (a law of large numbers applied to a martingale), for which we cite [26]. $\square$

In Chapter 3, PASTA becomes a key step of the Pollaczek–Khinchine formula in the form “the mean work an arriving turn finds waiting equals the time-average work waiting”.

1.5 The renewal-reward theorem

Definition 1.15 (Renewal process and rewards)

For i.i.d. positive random variables $C_1, C_2, \dots$ with $\mathbb{E}[C_1] < \infty$, let $\tau_n = C_1 + \dots + C_n$ and $M(t) = \#\{n \geq 1 : \tau_n \leq t\}$. M is a **renewal process** and C_n is its n -th **cycle**. Suppose each cycle carries a reward $R_n \geq 0$, the pairs (C_n, R_n) are i.i.d., and the cumulative reward $R(t)$ up to time t satisfies

$$\sum_{n=1}^{M(t)} R_n \leq R(t) \leq \sum_{n=1}^{M(t)+1} R_n$$

(this includes rewards that accrue gradually within a cycle). This is a **renewal-reward process**.

Theorem 1.16 (Renewal-reward theorem [22])

If $\mathbb{E}[R_1] < \infty$, then with probability one $\lim_{t \rightarrow \infty} \frac{R(t)}{t} = \frac{\mathbb{E}[R_1]}{\mathbb{E}[C_1]}$.

Proof. First we show $M(t)/t \rightarrow 1/\mathbb{E}[C_1]$ a.s. By definition $\tau_{M(t)} \leq t < \tau_{M(t)+1}$, and since each $\tau_n < \infty$, $M(t) \rightarrow \infty$. By the strong law, $\tau_n/n \rightarrow \mathbb{E}[C_1]$, so both ends of

$$\frac{\tau_{M(t)}}{M(t)} \leq \frac{t}{M(t)} < \frac{\tau_{M(t)+1}}{M(t)+1} \cdot \frac{M(t)+1}{M(t)}$$

converge to $\mathbb{E}[C_1]$. Now

$$\frac{1}{t} \sum_{n=1}^{M(t)} R_n = \frac{M(t)}{t} \cdot \frac{1}{M(t)} \sum_{n=1}^{M(t)} R_n \rightarrow \frac{1}{\mathbb{E}[C_1]} \mathbb{E}[R_1]$$

(the strong law for the second factor), and the upper bound has the same limit since $(M(t)+1)/t \rightarrow 1/\mathbb{E}[C_1]$. The conclusion follows by squeezing. $\square$

In the form “long-run time average = expected reward per cycle / expected cycle length”, the renewal-reward theorem is used directly for the residual service time (the inspection paradox) in Chapter 3.

1.6 Little’s law

Theorem 1.17 (Little’s law [15, 24])

For a sequence of customers (a_k, d_k) with $a_k \leq d_k < \infty$ and a_k nondecreasing, suppose the two limits

$$\lambda = \lim_{t \rightarrow \infty} \frac{A(t)}{t} \in (0, \infty), \quad W = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n W_k < \infty$$

exist. Then $L = \lim_{t \rightarrow \infty} \frac{1}{t} \int_0^t N(s) ds$ exists and

$$L = \lambda W. \tag{1.3}$$

Proof. Since $N(s) = \sum_k \mathbf{1}\{a_k \leq s < d_k\}$, we have $\int_0^t N(s) ds = \sum_k |[a_k, d_k) \cap [0, t]|$. Each term is at most W_k , is zero if $a_k > t$, and equals W_k exactly if $d_k \leq t$. Therefore

$$\sum_{k: d_k \leq t} W_k \leq \int_0^t N(s) ds \leq \sum_{k: a_k \leq t} W_k = \sum_{k=1}^{A(t)} W_k. \tag{1.4}$$

Upper bound: since $A(t) \rightarrow \infty$, $\frac{1}{t} \sum_{k \leq A(t)} W_k = \frac{A(t)}{t} \cdot \frac{1}{A(t)} \sum_{k \leq A(t)} W_k \rightarrow \lambda W$.

Lower bound: with $\bar{W}_n = \frac{1}{n} \sum_{k \leq n} W_k$ we have $\frac{W_n}{n} = \bar{W}_n - \frac{n-1}{n} \bar{W}_{n-1} \rightarrow W - W = 0$. Also $A(a_n) \geq n$ and $A(a_n^-) \leq n-1$, so $a_n/n \rightarrow 1/\lambda$, and therefore $W_n/a_n \rightarrow 0$. For any $\varepsilon > 0$ there is k_0 such that $d_k = a_k + W_k \leq (1 + \varepsilon)a_k$ for $k \geq k_0$. Then for $t \geq \max_{k < k_0} d_k$, every customer with $a_k \leq t/(1 + \varepsilon)$ has $d_k \leq t$, so

$$\frac{1}{t} \int_0^t N(s) ds \geq \frac{1}{t} \sum_{k=1}^{A(t/(1+\varepsilon))} W_k \rightarrow \frac{\lambda W}{1 + \varepsilon}.$$

Letting $\varepsilon \downarrow 0$, the liminf is at least λW . Together with the upper bound this gives (1.3). $\square$

The proof uses no distributional assumption and no service discipline. It is a deterministic theorem that holds on every sample path on which the limits exist. So depending on how we draw the boundary of “the system”, we obtain several identities.

Corollary 1.18 (Utilisation law [13])

Under the assumptions of Theorem 1.17, consider a single-server system and suppose the average of the service times $\frac{1}{n} \sum_{k \leq n} S_k$ converges to $\mathbb{E}[S]$.

- (i) The fraction of time the server is busy is $\rho = \lambda \mathbb{E}[S]$. In particular $\rho \leq 1$ in a stable system.
- (ii) For the waiting room, $L_q = \lambda W_q$, and under FIFO $L = L_q + \rho$.

Proof. (i) Take “the server” alone as the system. Customer k enters it when its service starts and leaves S_k later. Every customer is eventually served, so the arrival rate of this subsystem is also λ , and the customer average of its sojourn times is $\mathbb{E}[S]$. The number in this subsystem is 1 when the server is busy and 0 otherwise, so its time average is the fraction of time the server is busy. By Theorem 1.17 this equals $\lambda \mathbb{E}[S] = \rho$, and being a fraction it is at most one. (ii) Taking the waiting room alone as the system, the sojourn times are $W_{q,k}$, so $L_q = \lambda W_q$. Taking time averages in $N = N_q + (\text{number in service})$ gives $L = L_q + \rho$. $\square$

Example 1.19

Suppose $\lambda = 2$ turns per second arrive at a prefill instance and the mean prefill work is $\mathbb{E}[S] = 0.3$ seconds. By Corollary 1.18 the prefill server is busy a fraction $\rho_P = 0.6$ of the time. If the observed mean number of turns waiting for it is $L_q = 0.9$, then $W_q = L_q/\lambda = 0.45$ seconds, the mean TTFT (with no wait for memory) is $W_q + \mathbb{E}[S] = 0.75$ seconds, and the mean number of turns in the prefill stage (waiting or in service) is $L = \lambda \cdot 0.75 = 1.5$.

The scheduler’s objective is the mean number of turns in the stations, for example $L = L_P + L_D$ for the prefill server and the decode batch. By Theorem 1.17 this equals the total delay accumulated per unit time, $\lambda \cdot (\text{mean sojourn time})$, so “reduce the mean number of turns” and “reduce the mean delay” are the same goal. The “price of a miss” of Chapter 3 is defined precisely as a change in L_P .

1.7 Summary

- Prefill is a FIFO queue, decode a processor-sharing station, the tool call a delay station, and memory a constraint or a queue of slots.
- A Poisson process has independent exponential gaps; Poisson arrivals that do not anticipate the state see time averages (PASTA), but one session’s turns do not.
- Little’s law $L = \lambda W$ holds on every sample path on which the arrival rate and mean sojourn time exist, for any boundary of “the system”.

1.8 Exercises

Exercise 1.1. Prove Proposition 1.10. *Hint:* for (i) multiply the generating functions $\mathbb{E}[z^X] = e^{\lambda t(z-1)}$ of the independent increments; for (ii) condition on the total count $X = j + k$ in an interval, which splits binomially, and factor the joint law.

Exercise 1.2. For independent $X_1 \sim \text{Exp}(\mu_1)$ and $X_2 \sim \text{Exp}(\mu_2)$, show that $\mathbb{P}(X_1 < X_2) = \mu_1/(\mu_1 + \mu_2)$, and that the event $\{X_1 < X_2\}$ is independent of $\min(X_1, X_2)$. *Hint:* integrate $\mathbb{P}(X_1 < X_2, \min > t)$ against the density of X_1 .

Exercise 1.3. Each time a session makes a tool call, it stays away from the server for a mean of Z seconds. If tool calls start at rate λ , show that the time-average number of sessions in a tool call is λZ . Does this hold whatever the distribution of the tool-call duration? *Hint:* take the tool-call stage alone as “the system” and apply Theorem 1.17.

Chapter 2

Markovian Queues: M/M/1, Processor Sharing, and Insensitivity

Chapter 1 dealt with facts, such as the arrival process and Little's law, that depend hardly at all on the details of the model. This chapter treats the case where the state of the system (the number of jobs present) is a continuous-time Markov chain. The stationary distribution can then be computed exactly, and the model of the paper's decode stage (processor sharing) and its key property, *insensitivity*, come out of it.

2.1 Continuous-time Markov chains

Definition 2.1 (Continuous-time Markov chain)

A **generator** (Q -matrix) on a countable set $\mathcal{X}$ is a matrix $Q = (q(x, y))_{x, y \in \mathcal{X}}$ with $q(x, y) \geq 0$ for $x \neq y$, $q(x) := \sum_{y \neq x} q(x, y) < \infty$, and $q(x, x) = -q(x)$. $q(x, y)$ is the **transition rate** from x to y . Let $(Y_n)_{n \geq 0}$, the **jump chain**, be a discrete-time Markov chain on $\mathcal{X}$ with $\mathbb{P}(Y_{n+1} = y \mid Y_n = x) = q(x, y)/q(x)$ for $y \neq x$ when $q(x) > 0$, and $Y_{n+1} = x$ when $q(x) = 0$. Conditional on $(Y_n)_{n \geq 0}$, let the **holding times** $E_0, E_1, \dots$ be independent with $E_n \sim \text{Exp}(q(Y_n))$ ($E_n = \infty$ if $q(Y_n) = 0$), and let $J_0 = 0$, $J_{n+1} = J_n + E_n$. A process $(X_t)_{t \geq 0}$ with values in $\mathcal{X}$ is a **continuous-time Markov chain** (CTMC) with generator Q if it has the law of $X_t = Y_n$ for $J_n \leq t < J_{n+1}$. The chain is **non-explosive** if $\sup_n J_n = \infty$ almost surely, so that X_t is defined for every t , and **irreducible** if for all x, y there are states $x = x_0, x_1, \dots, x_k = y$ with $q(x_{i-1}, x_i) > 0$ for $i = 1, \dots, k$.

The memorylessness of the exponential distribution (Proposition 1.6) is what makes this definition possible. If holding times were not exponential, how long the chain has already stayed would affect the future and the Markov property would fail. Intuitively, $q(x, y)$ means that the probability of moving from x to y in a short time h is $q(x, y)h + o(h)$.

Definition 2.2 (Stationary distribution, balance equations)

A probability distribution π on $\mathcal{X}$ with $\pi Q = 0$, that is, for all x ,

$$\pi(x) q(x) = \sum_{y \neq x} \pi(y) q(y, x), \quad (2.1)$$

is a **stationary distribution**, and (2.1) are the **global balance equations**. The left side is the probability flow out of x , the right side the probability flow into x . The stronger condition

$$\pi(x) q(x, y) = \pi(y) q(y, x) \quad (\forall x \neq y) \quad (2.2)$$

is called **detailed balance**.

Lemma 2.3 (Detailed balance [19])

A probability distribution satisfying detailed balance is stationary.

Proof. Summing (2.2) over $y \neq x$ gives $\pi(x) \sum_{y \neq x} q(x, y) = \sum_{y \neq x} \pi(y) q(y, x)$, which is (2.1). $\square$

We quote the next theorem without proof [19, Ch. 3]. It lets us read the stationary probability $\pi(x)$ as “the fraction of time the state is x ”.

Theorem 2.4 (Ergodic theorem [19])

If an irreducible, non-explosive CTMC has a stationary distribution π , then π is unique, the chain is positive recurrent, and for any initial state and any f with $\sum_x |f(x)|\pi(x) < \infty$, almost surely

$$\frac{1}{t} \int_0^t f(X_s) ds \longrightarrow \sum_x f(x)\pi(x) \quad (t \rightarrow \infty).$$

2.2 Birth–death processes and M/M/1

Definition 2.5 (Birth–death process)

A CTMC on $\mathcal{X} = \{0, 1, 2, \dots\}$ whose only nonzero transition rates are $q(n, n+1) = \lambda_n > 0$ ($n \geq 0$) and $q(n, n-1) = \mu_n > 0$ ($n \geq 1$) is a **birth–death process**. λ_n are the birth rates and μ_n the death rates.

In a queue, n is the number of jobs in the system. An arrival increases n by one and a completion decreases it by one, so if arrivals and completions happen on exponential clocks, the number of jobs is a birth–death process. It is known that the process is non-explosive when the birth rates are bounded.

Theorem 2.6 (Stationary distribution of a birth–death process [13])

A birth–death process has a stationary distribution if and only if

$$H := \sum_{n=0}^{\infty} \prod_{k=1}^n \frac{\lambda_{k-1}}{\mu_k} < \infty \quad (2.3)$$

(an empty product is 1), and then the stationary distribution is $\pi(n) = H^{-1} \prod_{k=1}^n \lambda_{k-1} / \mu_k$.

Proof. (Necessity and the form of π .) Suppose π satisfies (2.1). At state 0, $\pi(0)\lambda_0 = \pi(1)\mu_1$. For $n \geq 1$, assume $\pi(n-1)\lambda_{n-1} = \pi(n)\mu_n$. Cancelling this term from the global balance at state n , $\pi(n)(\lambda_n + \mu_n) = \pi(n-1)\lambda_{n-1} + \pi(n+1)\mu_{n+1}$, gives $\pi(n)\lambda_n = \pi(n+1)\mu_{n+1}$. By induction, for all n

$$\pi(n)\lambda_n = \pi(n+1)\mu_{n+1}, \quad (2.4)$$

hence $\pi(n) = \pi(0) \prod_{k=1}^n \lambda_{k-1} / \mu_k$. For $\sum_n \pi(n) = 1$ we need $\pi(0) = 1/H > 0$, that is, $H < \infty$. (Sufficiency.) If $H < \infty$, the π defined above is a probability distribution, and (2.4) is exactly detailed balance (2.2), so π is stationary by Lemma 2.3. $\square$

Eq. (2.4) is called the **cut equation**: the flow across the boundary between $\{0, \dots, n\}$ and $\{n+1, \dots\}$ is the same in both directions.

The M/M/1 queue (Definition 1.4(a)) has Poisson arrivals of rate λ , service times $\text{Exp}(\mu)$ and one server, so its load is $\rho = \lambda/\mu$ (Definition 1.3).

Proposition 2.7 (M/M/1 [13])

In M/M/1, the number of jobs N_t (including the one in service) is a birth–death process with $\lambda_n = \lambda$, $\mu_n = \mu$. If $\rho < 1$, the stationary distribution is geometric, $\pi(n) = (1 - \rho)\rho^n$, and

$$L = \frac{\rho}{1 - \rho}, \quad W = \frac{1}{\mu - \lambda}, \quad W_q = \frac{\rho}{\mu(1 - \rho)}.$$

If $\rho \geq 1$, there is no stationary distribution. As $\rho \uparrow 1$, L , W and W_q all diverge to $+\infty$.

Proof. When $N_t = n \geq 1$, by memorylessness the time to the next arrival is $\text{Exp}(\lambda)$, the residual service time of the job in service is $\text{Exp}(\mu)$, and the two are independent. So the rate of $n \rightarrow n + 1$ is λ and that of $n \rightarrow n - 1$ is μ . In Theorem 2.6, $H = \sum_n \rho^n$ is finite only when $\rho < 1$, and then $H = 1/(1 - \rho)$. Hence $\pi(n) = (1 - \rho)\rho^n$ and $L = \sum_n n(1 - \rho)\rho^n = \rho/(1 - \rho)$. By Little’s law (Theorem 1.17), $W = L/\lambda = 1/(\mu - \lambda)$, and since the waiting time is the sojourn time minus the service time, $W_q = W - 1/\mu = \rho/(\mu(1 - \rho))$. $\square$

At $\rho = 0.9$ we get $L = 9$, and at $\rho = 0.99$, $L = 99$. This factor $1/(1 - \rho)$, which says that near capacity a small increase in mean load greatly increases the delay, reappears in the “price of a miss” of Chapter 3.

2.3 Multi-server queues and memory slots

An instance admits a turn only when its context fits in the instance’s memory. If every turn needs about the same memory, the pool is a set of identical *slots*, and the memory in front of an instance is a queue with several servers.

The model is the M/M/ m queue (Definition 1.4(b)): m identical servers, service times $\text{Exp}(\mu)$, and offered load $a = \lambda/\mu$, the mean number of busy servers.

Proposition 2.8 (M/M/ m and the Erlang C formula [13])

The number in system of M/M/ m is a birth–death process with $\lambda_n = \lambda$ and $\mu_n = \min(n, m)\mu$. If $a < m$, its stationary distribution is

$$\pi(n) = \pi(0) \frac{a^n}{n!} \quad (n \leq m), \quad \pi(n) = \pi(0) \frac{a^n}{m! m^{n-m}} \quad (n \geq m),$$

with $\pi(0)^{-1} = \sum_{k=0}^{m-1} a^k/k! + \frac{a^m}{m!} \frac{m}{m-a}$. The probability that an arriving customer waits is the **Erlang C formula**

$$C(m, a) = \sum_{n \geq m} \pi(n) = \pi(0) \frac{a^m}{m!} \frac{m}{m-a},$$

and the mean wait is $W_q = C(m, a)/(m\mu - \lambda)$. If $a \geq m$ there is no stationary distribution.

Proof. See Exercise 2.1. $\square$

For example, with $m = 4$ servers and offered load $a = 3$, $C(4, 3) \approx 0.51$: half of the arrivals wait, although only three quarters of the capacity is used; with $a = 2$ the share falls to 0.17. Waiting is a property of the whole distribution of busy servers, not of the mean load alone.

Memory slots. If every turn needs about c bytes of KV and the pool has M bytes, the pool holds $m = \lfloor M/c \rfloor$ slots. In an engine that admits a turn only when its context fits, memory is then a slot queue whose holding time runs from the start of a turn’s prefill to the end of its decode (Figure 1.1). Proposition 2.8 is exact for independent exponential holding times and Poisson arrivals. Real holding times are neither, so it is an approximation (the M/G/ m queue has no closed form; a common

correction multiplies W_q by $(1 + CV^2)/2$ of the holding time). With unequal contexts a turn needs several slots and the queue has no closed form. Cached blocks of paused sessions count as free: the cache does not remove slots but loses entries when an admission needs their blocks.

2.4 Processor sharing and state-dependent capacity

So far the server has processed one job at a time. The decode stage of an LLM does not.

Definition 2.9 (Work, processor sharing)

The **work** (service requirement) $S \geq 0$ of a job is the time the server needs to process it at full speed. Given a **capacity function** $\varphi : \{1, 2, \dots\} \rightarrow (0, \infty)$, the rule under which, when n jobs are present, the server processes $\varphi(n)$ units of work per second and splits it equally among the n jobs, so that each job's remaining work decreases at rate $\varphi(n)/n$, is called **processor sharing** (PS). $\varphi \equiv 1$ is classical PS; with Poisson arrivals and a general work distribution G we write **M/G/1-PS**. The load is $\rho = \lambda \mathbb{E}[S]$.

In LLM serving

In continuous batching, one decode step advances every request in the batch by one token. We therefore model the decode stage as PS, with the m requests sharing the device equally. Because the weights are read once for the whole batch, $\varphi(m)$ increases with m , and because memory (the KV cache) limits the batch size, it eventually saturates. The decode stage of the paper is exactly this model: the work is the decode demand D_i and the load is $\rho_D = \lambda \mathbb{E}[D]$.

Proposition 2.10 (PS with exponential work [S])

For a PS server with capacity φ , Poisson arrivals of rate λ , and exponential work with mean $\mathbb{E}[S]$, the number of jobs is a birth–death process with $\lambda_n = \lambda$ and $\mu_n = \varphi(n)/\mathbb{E}[S]$. Hence, with $F(n) := \prod_{k=1}^n \varphi(k)$, if $G(\rho) := \sum_{n \geq 0} \rho^n / F(n) < \infty$ then

$$\pi(n) = \frac{\rho^n / F(n)}{G(\rho)}, \quad \rho = \lambda \mathbb{E}[S]. \quad (2.5)$$

Proof. In state n , by memorylessness the remaining works of the n jobs are independent, each $\text{Exp}(1/\mathbb{E}[S])$, and each decreases at speed $\varphi(n)/n$. Hence each job's completion time is exponential with rate $\varphi(n)/(n\mathbb{E}[S])$, and the first of the n completions is exponential with rate $\varphi(n)/\mathbb{E}[S]$ (Lemma 1.7). The rest follows from Theorem 2.6, since $\prod_{k \leq n} \lambda \mathbb{E}[S] / \varphi(k) = \rho^n / F(n)$. $\square$

With $\varphi \equiv 1$, (2.5) is the geometric distribution of M/M/1. With exponential work, the service order does not change the distribution of the number of jobs.

Example 2.11 (Saturating capacity)

Let $\varphi(m) = \min(m, \bar{m})$. Below $\bar{m}$ the throughput grows in proportion to the batch size, and it saturates at $\bar{m}$ (the simplest model of a memory that holds only $\bar{m}$ requests). This is the same birth–death process as M/M/ $\bar{m}$ with $\bar{m}$ servers, and by (2.5), $\pi(n) \propto \rho^n / n!$ for $n \leq \bar{m}$ and $\pi(n) \propto \rho^n / (\bar{m}! \bar{m}^{n-\bar{m}})$ for $n \geq \bar{m}$. $G(\rho) < \infty$ if and only if $\rho < \bar{m} = \sup \varphi$. If the load exceeds the saturated capacity, the system is unstable however large the batch grows.

The next theorem says that more expected demand never leaves fewer jobs. This sounds obvious, but it is not true in general: it depends on the discipline.

More work, fewer jobs. At a single FIFO server compare two work laws with Poisson arrivals at $\lambda = 0.5$. Under law A a job takes 0.1 with probability 0.9 and 9.1 with probability 0.1 ($\rho = 0.5$); under

law B every job takes 1.2 ($\rho = 0.6$), so B carries 20 % more work. The Pollaczek–Khinchine formula of Chapter 3 gives a mean number in system of 2.57 under A and 1.05 under B: one long job of law A holds the server while short jobs pile up behind it. Under processor sharing with $\varphi \equiv 1$ the mean number is $\rho/(1 - \rho)$ for every work law (Theorem 2.14 below), 1.0 under A and 1.5 under B, in the order of the load. Theorem 2.12 together with insensitivity makes this true for every capacity function φ .

Theorem 2.12 (Monotonicity of the mean batch size)

Let φ be any positive function and R the radius of convergence of the power series $G(\rho) = \sum_n \rho^n / F(n)$. The mean $L(\rho) = \sum_n n \pi(n)$ of the stationary distribution (2.5) is differentiable on $(0, R)$ and

$$\frac{dL}{d\rho} = \frac{\text{Var}_\rho(N)}{\rho} \geq 0.$$

Hence L is nondecreasing in the load. If φ is nondecreasing, then $R = \sup_n \varphi(n)$.

Proof. Inside its radius of convergence a power series can be differentiated term by term, and the differentiated series has the same radius (a standard theorem of analysis). With $a_n = 1/F(n)$ we have $\rho G'(\rho) = \sum_n n a_n \rho^n$ and $\rho^2 G''(\rho) = \sum_n n(n-1) a_n \rho^n$, so

$$L = \frac{\rho G'}{G}, \quad \mathbb{E}_\rho[N(N-1)] = \frac{\rho^2 G''}{G}.$$

By the quotient rule,

$$\rho \frac{dL}{d\rho} = \frac{\rho G'}{G} + \frac{\rho^2 G''}{G} - \left(\frac{\rho G'}{G} \right)^2 = \mathbb{E}[N] + \mathbb{E}[N(N-1)] - \mathbb{E}[N]^2 = \text{Var}(N) \geq 0.$$

If φ is nondecreasing with $\varphi_\infty := \sup \varphi$, then $a_{n+1}/a_n = 1/\varphi(n+1) \rightarrow 1/\varphi_\infty$, so by the ratio test $R = \varphi_\infty$ ($R = \infty$ if $\varphi_\infty = \infty$). $\square$

For $\rho > 0$, N is not degenerate, so in fact L is strictly increasing. The theorem says, without knowing anything about the shape of φ , that “a policy that increases the mean demand does not shrink the batch”. Since φ must be calibrated from measurements, such shape-free conclusions matter.

2.5 Insensitivity

Proposition 2.10 assumed exponential work. But the decode demand is proportional to the output length, and there is no reason for it to be exponential. Remarkably, under PS the conclusion does not depend on the work distribution.

Definition 2.13 (Insensitivity)

A stationary quantity of a queue (here, the distribution of the number of jobs) is **insensitive** to the work distribution G if it depends on G only through its mean $\int x dG(x)$.

Theorem 2.14 (Insensitivity of PS [8, 3])

For a PS server with capacity φ and Poisson arrivals of rate λ , the stationary distribution of the number of jobs is insensitive to the work distribution and equals (2.5). We prove it here when the work is a finite mixture of exponentials, that is, $\text{Exp}(\mu_k)$ with probability α_k ($k = 1, \dots, K$, $\sum_k \alpha_k = 1$).

Proof. Think of each job as independently drawing a type k with probability α_k on arrival and then drawing its work from $\text{Exp}(\mu_k)$. The vector of job counts by type, $\mathbf{n} = (n_1, \dots, n_K)$ with $n = |\mathbf{n}| = \sum_k n_k$, is a CTMC. By thinning of the Poisson process (Proposition 1.10(ii)), type k arrives at rate

$\lambda\alpha_k (\mathbf{n} \rightarrow \mathbf{n} + e_k)$, and as in the proof of Proposition 2.10 a single type- k job finishes at rate $\mu_k\varphi(n)/n$, so the rate of $\mathbf{n} \rightarrow \mathbf{n} - e_k$ is $n_k\mu_k\varphi(n)/n$. Let $\rho_k = \lambda\alpha_k/\mu_k$. We show that

$$\pi(\mathbf{n}) = B \frac{n!}{n_1! \cdots n_K!} \frac{\prod_k \rho_k^{n_k}}{F(n)}$$

satisfies global balance. If $n_k \geq 1$, then $\pi(\mathbf{n})/\pi(\mathbf{n} - e_k) = \frac{n}{n_k} \cdot \frac{\rho_k}{\varphi(n)}$, so

$$\pi(\mathbf{n}) \frac{n_k\mu_k\varphi(n)}{n} = \pi(\mathbf{n} - e_k) \rho_k\mu_k = \pi(\mathbf{n} - e_k) \lambda\alpha_k. \quad (2.6)$$

This says “the flow out of $\mathbf{n}$ by a type- k completion equals the flow into $\mathbf{n}$ by a type- k arrival”. Applying (2.6) at $\mathbf{n} + e_k$ and summing over k ,

$$\sum_k \pi(\mathbf{n} + e_k) \frac{(n_k + 1)\mu_k\varphi(n + 1)}{n + 1} = \sum_k \pi(\mathbf{n}) \lambda\alpha_k = \pi(\mathbf{n}) \lambda,$$

that is, “the flow out by arrivals equals the flow in by completions”. Adding the two gives global balance at $\mathbf{n}$. Splitting global balance in this way is called **partial balance**. Now, by the multinomial theorem,

$$\mathbb{P}(N = n) = \sum_{|\mathbf{n}|=n} \pi(\mathbf{n}) = \frac{B}{F(n)} \sum_{|\mathbf{n}|=n} \frac{n!}{\prod_k n_k!} \prod_k \rho_k^{n_k} = \frac{B \rho^n}{F(n)}, \quad \rho = \sum_k \rho_k = \lambda \sum_k \frac{\alpha_k}{\mu_k} = \lambda \mathbb{E}[S].$$

The normalisability condition is $G(\rho) < \infty$, with $B = 1/G(\rho)$. This is (2.5), and G involves only $\mathbb{E}[S]$. (Uniqueness follows from irreducibility and Theorem 2.4.) $\square$

Remark 2.15

Insensitivity for general work distributions is due to [8] and [3] (BCMP). One approach to a proof is to extend the partial balance argument above to phase-type distributions and take limits, using the fact that phase-type distributions are dense, under weak convergence, among distributions on $[0, \infty)$. Here we only cite it. The same insensitivity holds for a **delay station** (an infinite-server station where every job immediately gets its own server, so nobody waits), which we use in Chapter 4.

FIFO is not insensitive. Compare three service-time distributions with the same mean $\mathbb{E}[S] = 1$ and the same load $\rho = 0.5$ ($\lambda = 0.5$). Borrowing from Chapter 3 the Pollaczek–Khinchine formula $W_q = \lambda \mathbb{E}[S^2]/(2(1 - \rho))$ and Little’s law $L = \lambda W_q + \rho$, we get the following.

Service time S	$\mathbb{E}[S^2]$	FIFO L	PS L
constant 1	1	0.75	1
Exp(1)	2	1.00	1
0.1 w.p. 0.9, 9.1 w.p. 0.1	8.29	2.57	1

Under FIFO one long job blocks every job behind it (head-of-line blocking). Under PS short jobs share capacity with the long one and leave first. The fairness of PS is also visible job by job: in M/G/1-PS, the mean sojourn time of a job with work x is $x/(1 - \rho)$, proportional to its own size and independent of the size distribution of the other jobs [12]. This difference separates the two stages of the paper: the FIFO prefill queue sees the second moment of the work, while the PS decode stage sees only the mean.

2.6 The price at the decode stage

Proposition 2.16 (The decode stage prices only work)

For a PS station with constant capacity C and load $\rho < C$, $L = \rho/(C - \rho)$. If a policy change increases the work of a fraction q_i of arrivals by $\Delta D_i \geq 0$, raising the load to $\rho' = \rho + \lambda \sum_i q_i \Delta D_i < C$, then exactly

$$\Delta L = \frac{C(\rho' - \rho)}{(C - \rho)(C - \rho')} = \frac{C - \rho}{C - \rho'} \lambda \sum_i q_i \Phi_i^D, \quad \Phi_i^D = \frac{C \Delta D_i}{(C - \rho)^2}.$$

In particular $\lambda \sum_i q_i \Phi_i^D \leq \Delta L$.

Proof. With $\varphi \equiv C$ we have $F(n) = C^n$, and (2.5) is geometric with ratio ρ/C , so $L = (\rho/C)/(1 - \rho/C) = \rho/(C - \rho)$. Therefore

$$\Delta L = \frac{\rho'}{C - \rho'} - \frac{\rho}{C - \rho} = \frac{\rho'(C - \rho) - \rho(C - \rho')}{(C - \rho)(C - \rho')} = \frac{C(\rho' - \rho)}{(C - \rho)(C - \rho')}.$$

Substituting $\rho' - \rho = \lambda \sum_i q_i \Delta D_i$ and multiplying and dividing by $(C - \rho)^2$ gives the second equality. Since $\rho' \geq \rho$, the leading factor is at least one. $\square$

$\Phi_i^D = \Delta D_i L'(\rho)$ is the first-order approximation, and the factor $(C - \rho)/(C - \rho')$ corrects for curvature. For example, with $C = 1$ and $\rho = 0.5 \rightarrow 0.6$, $\Delta L = 0.5$, the first-order term is 0.4, and the factor is 1.25.

Corollary 2.17

If a policy change alters the distribution of the decode work but not its mean $\mathbb{E}[D]$, the distribution of the number of jobs at the decode stage does not change.

Proof. By Theorem 2.14 the distribution depends only on $\rho_D = \lambda \mathbb{E}[D]$. $\square$

In LLM serving

KV-cache hits and misses change only the prefill work; they do not change the decode demand $D_i = o_i(\beta K_i + \omega/m)$ of Section 5.1. So by Corollary 2.17 a miss costs nothing at the decode stage to first order, and its whole price is paid at the FIFO prefill queue (Chapter 3), on an engine whose decode capacity does not depend on the prefill load (Section 5.5 says what an exclusive prefill step changes). Only decisions that add decode demand, such as admitting a session, pay the load factor of Proposition 2.16. Proposition 2.16 and Corollary 2.17 are the content of the paper's proposition "The decode stage prices only work".

2.7 Summary

- A birth–death process has an explicit stationary law: M/M/1, and M/M/ m (Erlang C) for memory slots.
- Under processor sharing the number of jobs depends on the work only through its mean, and its mean grows with the load.
- So the decode stage prices only work, and a miss costs nothing there.

2.8 Exercises

Exercise 2.1. Prove Proposition 2.8. *Hint:* with n customers the first completion comes at rate $\min(n, m)\mu$ (Lemma 1.7); apply Theorem 2.6, use PASTA for the waiting probability, and compute $L_q = \sum_{n \geq m} (n - m)\pi(n)$ before applying Little's law.

Exercise 2.2. For $\varphi(n) = n$ (each job has its own server, M/M/ ∞), show that (2.5) is the Poisson distribution with mean ρ , and check the formula $dL/d\rho = \text{Var}(N)/\rho$ of Theorem 2.12 directly. *Hint:* $F(n) = n!$, and a Poisson distribution has equal mean and variance.

Exercise 2.3. Show that in M/M/1-FIFO the sojourn time of an arriving job is $\text{Exp}(\mu - \lambda)$. *Hint:* by PASTA (Theorem 1.14) the number found on arrival is distributed as $(1 - \rho)\rho^n$, and conditionally the sojourn time is a sum of $n + 1$ independent $\text{Exp}(\mu)$ variables.

Exercise 2.4 (One fast server or two slow ones). Compare M/M/1 with service rate 2μ and M/M/2 with rate μ per server, both at arrival rate $\lambda < 2\mu$. Show that $C(2, a) = a^2/(2 + a)$ with $a = \lambda/\mu$, and that the single fast server has the smaller mean sojourn time but the larger mean wait W_q . What does this say about splitting a memory pool into slots? *Hint:* use Proposition 2.8 for M/M/2 and Proposition 2.7 for M/M/1.

Chapter 3

The M/G/1 Queue and the Pollaczek–Khinchine Formula: the Price of a Miss

In Chapter 2 the service times were exponential, so the state of the queue was summarised by a single number, the number of customers. Prefill work is not exponential. It is a general distribution that mixes short appends (hits) with long recomputations (misses). The goal of this chapter is to compute the mean waiting time of a FIFO queue with a general service distribution (the Pollaczek–Khinchine formula), and then to differentiate it to prove the central proposition of the paper, the price of a miss. The key message is a single one. *The mean wait of a FIFO queue depends not only on the mean of the work but also on its second moment.*

3.1 The M/G/1 queue and the residual service time

The M/G/1 queue (Definition 1.4(c)) has Poisson arrivals of rate $\lambda > 0$, i.i.d. work S_k with an arbitrary law G (“General”), independent of the arrivals, with generic S and $\mathbb{E}[S] < \infty$, and one FIFO server that serves each customer to completion without interruption. The load is $\rho = \lambda \mathbb{E}[S]$ (Definition 1.3). As in Definition 1.1, W_q is the wait from arrival to the start of service, the sojourn time is $W_q + S$, and L_q and L are the time averages of the number of customers in the waiting room and in the whole system.

Unlike the exponential distribution, a general service distribution is not memoryless (Proposition 1.6). How much longer a customer who has already been served for 3 seconds will take depends on the information “3 seconds”. The wait of an arriving customer therefore requires the quantity “how much service is left for the customer now in service”.

Definition 3.1 (Residual service time)

If the server is serving a customer at time t , let $R(t)$ be the remaining service time of that customer; if the server is idle, set $R(t) = 0$. $R(t)$ is called the **residual service time**.

The sample path of $R(t)$ is a sawtooth. At the moment service of customer k begins, it jumps up to S_k ; it then decreases with slope -1 , and when it reaches 0 the next service begins or the server becomes idle. Hence the contribution of a single customer k to $\int R(t) dt$ is the area $S_k^2/2$ of a triangle with base and height S_k .

Lemma 3.2 (Time average of the residual service time [13])

In an M/G/1 queue, suppose $\mathbb{E}[S^2] < \infty$ and that the number $D(T)$ of customers who have completed service by time T satisfies $D(T)/T \rightarrow \lambda$ (a.s.) (stability; this holds if $\rho < 1$). Then almost surely

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T R(t) dt = \frac{\lambda \mathbb{E}[S^2]}{2} = \rho \cdot \frac{\mathbb{E}[S^2]}{2\mathbb{E}[S]}.$$

In particular, averaged only over the time the server is busy, the mean residual service time is $\mathbb{E}[S^2]/(2\mathbb{E}[S])$.

Proof. Let $A(T)$ be the number of arrivals in $[0, T]$. The triangles of the customers who completed service in $[0, T]$ lie entirely in $[0, T]$, and every customer whose service has started arrived before T , so

$$\sum_{k=1}^{D(T)} \frac{S_k^2}{2} \leq \int_0^T R(t) dt \leq \sum_{k=1}^{A(T)} \frac{S_k^2}{2}.$$

(Under FIFO the service order equals the arrival order, so the first $D(T)$ customers are exactly those who have completed service.) By the strong law of large numbers $\frac{1}{n} \sum_{k \leq n} S_k^2 \rightarrow \mathbb{E}[S^2]$; for a Poisson process $A(T)/T \rightarrow \lambda$, and by assumption $D(T)/T \rightarrow \lambda$. Hence

$$\frac{1}{T} \sum_{k=1}^{D(T)} \frac{S_k^2}{2} = \frac{D(T)}{T} \cdot \frac{1}{D(T)} \sum_{k=1}^{D(T)} \frac{S_k^2}{2} \rightarrow \lambda \cdot \frac{\mathbb{E}[S^2]}{2},$$

and the upper sum has the same limit. The squeeze gives the first equality. The second equality is $\lambda = \rho/\mathbb{E}[S]$. The fraction of time the server is busy is ρ (Corollary 1.18) and $R = 0$ during idle time, so the average over busy time is $\lambda \mathbb{E}[S^2]/(2\rho) = \mathbb{E}[S^2]/(2\mathbb{E}[S])$. $\square$

Remark 3.3 (Inspection paradox)

$\mathbb{E}[S^2]/(2\mathbb{E}[S]) = \frac{\mathbb{E}[S]}{2}(1 + \text{CV}^2)$, where $\text{CV}^2 = \text{Var}[S]/\mathbb{E}[S]^2$ is the **squared coefficient of variation**. The intuition “a customer in service has, on average, about half of its service left” fails because, when we look at the server at a random time, the probability of catching a long service is proportional to its length. Indeed, the service observed at a busy instant has the **length-biased distribution** with density $x G(dx)/\mathbb{E}[S]$, whose mean is $\mathbb{E}[S^2]/\mathbb{E}[S]$; the observation instant is uniform within that service, so the mean remaining time is half of it. This is the **inspection paradox**. For the exponential distribution $\text{CV}^2 = 1$, so the mean residual equals $\mathbb{E}[S]$ itself, in agreement with memorylessness.

3.2 The Pollaczek–Khinchine formula

Theorem 3.4 (Pollaczek–Khinchine mean formula [20, 9])

In an M/G/1 FIFO queue let $\rho < 1$ and $\mathbb{E}[S^2] < \infty$. Then in steady state

$$\mathbb{E}[W_q] = \frac{\lambda \mathbb{E}[S^2]}{2(1 - \rho)}, \quad L = \rho + \frac{\lambda^2 \mathbb{E}[S^2]}{2(1 - \rho)}. \quad (3.1)$$

Steady-state assumption. The proof takes the following facts as given. If $\rho < 1$, the queue is stable, the waiting time $W_q^{(k)}$ of the k -th customer converges in distribution, and the limit law has a finite mean (when $\mathbb{E}[S^2] < \infty$). Moreover the customer averages and the time averages exist almost surely. These are standard results obtained by analysing the Lindley recursion $W_q^{(k+1)} = (W_q^{(k)} + S_k - \tau_{k+1})^+$ (τ the interarrival times); see [2, Ch. 3, 10] and [13, Ch. 5]. The proof below accepts this existence and computes the *value* of the limit.

Proof. Consider a customer who has just arrived in steady state (the “tagged customer”). Under FIFO this customer waits for (1) the residual service time of the customer in service and (2) the service times of the N_q customers already in the waiting room:

$$W_q = R + \sum_{j=1}^{N_q} S_{(j)},$$

where R is the residual service time at the arrival instant and $S_{(j)}$ is the service time of the j -th waiting customer.

Mean of (1). Poisson arrivals see time averages (PASTA, Theorem 1.14). So the law of R seen at the arrival instant is the time-stationary law of $R(t)$, and by Lemma 3.2 $\mathbb{E}[R] = \lambda \mathbb{E}[S^2]/2$.

Mean of (2). The waiting customers have not started service yet. Under FIFO the state of the queue up to the arrival of the tagged customer is determined only by the arrival times and the service times of the customers who have already started service. Hence the service times $S_{(1)}, S_{(2)}, \dots$ of the customers not yet started can be regarded as i.i.d. G random variables independent of N_q . Therefore $\mathbb{E}[\sum_{j \leq N_q} S_{(j)}] = \sum_n \mathbb{P}(N_q = n) n \mathbb{E}[S] = \mathbb{E}[N_q] \mathbb{E}[S]$. By PASTA again, $\mathbb{E}[N_q]$ equals the time average L_q of the number in the waiting room.

Combining. Applying Little’s law (Theorem 1.17) to the waiting room alone as the “system” gives $L_q = \lambda \mathbb{E}[W_q]$. Therefore

$$\mathbb{E}[W_q] = \frac{\lambda \mathbb{E}[S^2]}{2} + \lambda \mathbb{E}[W_q] \mathbb{E}[S] = \frac{\lambda \mathbb{E}[S^2]}{2} + \rho \mathbb{E}[W_q].$$

Since $\mathbb{E}[W_q] < \infty$, subtracting $\rho \mathbb{E}[W_q]$ from both sides and dividing by $1 - \rho > 0$ gives the first formula. The second formula is Little’s law applied to the whole system: $L = \lambda(\mathbb{E}[W_q] + \mathbb{E}[S]) = \rho + \lambda \mathbb{E}[W_q]$. $\square$

Corollary 3.5 (Variance increases the wait [13])

Under the assumptions of Theorem 3.4,

$$\mathbb{E}[W_q] = \frac{\rho}{1 - \rho} \cdot \frac{1 + \text{CV}^2}{2} \cdot \mathbb{E}[S]. \quad (3.2)$$

Hence, of two work laws with the same λ and the same mean $\mathbb{E}[S]$ (so the same load $\rho < 1$), the one with the larger variance waits strictly longer. In particular:

- (i) M/M/1 ($\text{CV}^2 = 1$): $\mathbb{E}[W_q] = \rho \mathbb{E}[S]/(1 - \rho)$, in agreement with Proposition 2.7.
- (ii) M/D/1 (deterministic service, $\text{CV}^2 = 0$): $\mathbb{E}[W_q] = \rho \mathbb{E}[S]/(2(1 - \rho))$, exactly half of M/M/1 at the same load.

Proof. See Exercise 3.1. $\square$

Remark 3.6

Contrast this with the PS queue of Chapter 2 (Definition 2.9). The mean number in an M/G/1-PS queue is $\rho/(1 - \rho)$, insensitive to the work law (Theorem 2.14), while the mean number in FIFO, Eq. (3.1), is sensitive to the law through $\mathbb{E}[S^2]$. A single long job blocks every customer behind it under FIFO, but not under PS, where all customers share the server.

3.3 Mixtures of hits and misses

The KV cache policy decides whether each turn is a hit or a miss, and hence sets the law of the prefill work.

Let S^{hit} and S^{miss} be square-integrable nonnegative random variables with $S^{\text{miss}} \geq S^{\text{hit}}$ (a.s.), and let U be uniform on $(0, 1)$ and independent of them. At **hit rate** $h \in [0, 1]$ the work is the **hit/miss mixture**

$$S_h = \mathbf{1}\{U \leq h\} S^{\text{hit}} + \mathbf{1}\{U > h\} S^{\text{miss}}, \quad (3.3)$$

which has the law of S^{hit} with probability h and that of S^{miss} with probability $1 - h$.

Corollary 3.7 (A higher hit rate lowers the PK wait)

Let $m_1(h) = \mathbb{E}[S_h]$, $m_2(h) = \mathbb{E}[S_h^2]$ and $\rho(h) = \lambda m_1(h)$.

- (i) m_1 , m_2 and ρ are affine and nonincreasing in h .
- (ii) If $\rho(h_0) < 1$, then $\rho(h) < 1$ for all $h \in [h_0, 1]$, and $W(h) = \lambda m_2(h)/(2(1 - \rho(h)))$ is nonincreasing on $[h_0, 1]$.

Proof. See Exercise 3.2. □

3.4 The price of a miss

Corollary 3.7 gives only a direction. A scheduler needs a *magnitude*. If we drop the KV state of a particular program i , so that its next turn is a miss, by how much does the delay of the whole prefill stage grow?

Model the prefill stage as an M/G/1 FIFO queue with arrival rate λ , work S ($m_1 = \mathbb{E}[S]$, $m_2 = \mathbb{E}[S^2] < \infty$) and load $\rho = \lambda m_1 < 1$. Let $W = \lambda m_2/(2(1 - \rho))$ be the PK wait and L_P the mean number in the prefill stage (waiting and in service). By Little's law

$$L_P = \lambda(W + m_1) = \rho + \frac{\lambda^2 m_2}{2(1 - \rho)}. \quad (3.4)$$

For programs (classes) $i = 1, \dots, k$, a **policy change** turns a fraction $q_i \geq 0$ of arrivals ($\sum_i q_i \leq 1$) from hits with work $S_i^{\text{hit}} \geq 0$ into misses with work $S_i^{\text{miss}} = S_i^{\text{hit}} + \Delta S_i$, $\Delta S_i \geq 0$. The moments of the work become

$$m'_1 = m_1 + \delta_1, \quad \delta_1 = \sum_i q_i \Delta S_i, \quad m'_2 = m_2 + \delta_2, \quad \delta_2 = \sum_i q_i ((S_i^{\text{miss}})^2 - (S_i^{\text{hit}})^2),$$

and the new load is $\rho' = \rho + \lambda \delta_1$. We write L'_P for the new mean number and $\Delta L_P = L'_P - L_P$ for the change.

Theorem 3.8 (The price of a miss)

In this setting, suppose $\rho' < 1$ and define

$$\Phi_i = \Delta S_i + \frac{\lambda((S_i^{\text{miss}})^2 - (S_i^{\text{hit}})^2)}{2(1 - \rho)} + \frac{\lambda W \Delta S_i}{1 - \rho}. \quad (3.5)$$

Then

$$\lambda \sum_i q_i \Phi_i \leq \Delta L_P \leq \frac{1 - \rho}{1 - \rho'} \lambda \sum_i q_i \Phi_i.$$

Proof. Write $u = 1 - \rho > 0$. Then $1 - \rho' = u - \lambda \delta_1 > 0$, and since $\delta_1 \geq 0$, $0 < u - \lambda \delta_1 \leq u$. Applying the PK formula (Theorem 3.4) to the new work,

$$\Delta L_P = \lambda \delta_1 + \frac{\lambda^2}{2} \left(\frac{m_2 + \delta_2}{u - \lambda \delta_1} - \frac{m_2}{u} \right) = \lambda \delta_1 + \frac{\lambda^2}{2} \cdot \frac{u(m_2 + \delta_2) - (u - \lambda \delta_1)m_2}{u(u - \lambda \delta_1)} = \lambda \delta_1 + \frac{Y}{u(u - \lambda \delta_1)},$$

where $Y = \frac{\lambda^2}{2}(u\delta_2 + \lambda\delta_1 m_2) \geq 0$. On the other hand, taking the q_i -weighted sum of Eq. (3.5) and using $W = \lambda m_2/(2u)$,

$$\lambda \sum_i q_i \Phi_i = \lambda\delta_1 + \frac{\lambda^2\delta_2}{2u} + \frac{\lambda^2 W \delta_1}{u} = \lambda\delta_1 + \frac{\lambda^2}{2u^2}(u\delta_2 + \lambda\delta_1 m_2) = \lambda\delta_1 + \frac{Y}{u^2}.$$

Lower bound. Since $u - \lambda\delta_1 \leq u$, $Y/(u(u - \lambda\delta_1)) \geq Y/u^2$, and hence $\Delta L_P \geq \lambda \sum_i q_i \Phi_i$.

Upper bound. A direct computation gives

$$\frac{u}{u - \lambda\delta_1} \left(\lambda\delta_1 + \frac{Y}{u^2} \right) - \Delta L_P = \lambda\delta_1 \left(\frac{u}{u - \lambda\delta_1} - 1 \right) + \frac{Y}{u(u - \lambda\delta_1)} - \frac{Y}{u(u - \lambda\delta_1)} = \frac{(\lambda\delta_1)^2}{u - \lambda\delta_1} \geq 0.$$

Since $u/(u - \lambda\delta_1) = (1 - \rho)/(1 - \rho')$, the upper bound holds. $\square$

Reading the three terms. Φ_i is measured in seconds. Misses of program i occur at rate λq_i , so by Little's law $\lambda q_i \Phi_i$ is an increase in the mean number, and Φ_i is “the total waiting time, summed over all turns, that one miss adds”.

- (1) ΔS_i : the miss itself takes longer to serve (its own cost).
- (2) $\lambda((S_i^{\text{miss}})^2 - (S_i^{\text{hit}})^2)/(2(1 - \rho))$: turns that arrive while the miss is being served wait for all of it. This comes from the second moment in the PK formula, and its size is the difference of the *squares* of the work. It is the cost imposed on other customers, the **externality** of the economics of queues; in these notes we call it the **head-of-line term**.
- (3) $\lambda W \Delta S_i/(1 - \rho)$: the extra work raises the load, which stretches every wait by the factor $1/(1 - \rho)$ (the load term).

Since $(S^{\text{miss}})^2 - (S^{\text{hit}})^2 = (S^{\text{miss}} + S^{\text{hit}})\Delta S$,

$$\frac{(2)}{(1)} = \frac{\lambda(S_i^{\text{miss}} + S_i^{\text{hit}})}{2(1 - \rho)}, \quad \frac{(2)}{(3)} = \frac{S_i^{\text{miss}} + S_i^{\text{hit}}}{2W}.$$

The first ratio is of the order of “the number of turns that arrive while one miss is served”. When it is small, the price is close to ΔS_i ; when it is large, the head-of-line term dominates. Once the mean wait W exceeds half the sum of the miss and hit work, the load term exceeds the head-of-line term.

Example 3.9 (At moderate load the head-of-line term dominates)

In seconds, let $\lambda = 1.25$, $m_1 = 0.4$, $m_2 = 0.5$, so $\rho = 0.5$ and $W = 0.625$. Let a program have hit work $S^{\text{hit}} = 0.2$ and miss work $S^{\text{miss}} = 2.0$ ($\Delta S = 1.8$). The three terms are

$$\Delta S = 1.8, \quad \frac{1.25(4 - 0.04)}{2 \cdot 0.5} = 4.95, \quad \frac{1.25 \cdot 0.625 \cdot 1.8}{0.5} = 2.8125,$$

so $\Phi = 9.5625$ seconds. One miss creates more than five times its own extra service in total waiting, and most of it is paid by the turns queued behind it. The two ratios are 2.75 and 1.76. If a fraction $q = 0.01$ of this program's turns turn into misses, then $\rho' = 0.5225$, and the exact change $\Delta L_P \approx 0.1241$ lies in the interval $[0.1195, 0.1252]$ of the theorem.

Corollary 3.10 (Divergence near saturation)

Fix the work law ($m_1 > 0$, $m_2 > 0$) and $\Delta S_i > 0$, and let $\lambda \uparrow 1/m_1$ (that is, $\rho \uparrow 1$). Then $\Phi_i \rightarrow \infty$.

Proof. See Exercise 3.3. $\square$

Corollary 3.11 (Monotonicity in the context length)

Let the prefill cost be $P(n, K) = an + bn(K + n/2)$ ($a, b \geq 0$; the cost of computing n tokens on top of a prefix of length K). A hit costs $S^{\text{hit}} = P(n, K)$ and a miss, which recomputes the whole context, costs $S^{\text{miss}} = P(K + n, 0)$. Then $\Delta S = aK + bK^2/2$, and with λ, ρ, W and the number n of appended tokens fixed, Φ is nondecreasing in $K \geq 0$.

Proof. See Exercise 3.4. □

In LLM serving

Chunked prefill interleaves chunks of a long prefill with decode steps, but pending prefills are still served in arrival order: a prefill behind a long miss waits for every chunk of it, as long as the oldest prefill may take all the budget the decode batch leaves. Then the head-of-line term (2) remains, whatever the chunk size; a per-prefill cap lets later prefills bypass the miss (Exercise 5.3). Corollary 3.11 says a longer context has a larger price, but the scheduler compares $p_i \Phi_i / c_i$, and that order can flip with the load (Example 5.6).

Remark 3.12 (Limits of the M/G/1 assumption)

A pool holds a few dozen live sessions, and a session waiting for prefill cannot create its next turn, so arrivals slow down when the queue is long. For exponential work the finite-population model of Chapter 4 shows that the open wait is an upper bound (Proposition 4.4); for general work the comparison is a question for measurement.

3.5 Summary

- The FIFO wait depends on the second moment of the work, $\mathbb{E}[W_q] = \lambda \mathbb{E}[S^2] / (2(1 - \rho))$.
- The price of a miss has an own-work, a head-of-line and a load term, and brackets the change in the mean number at prefill.
- It diverges near saturation and grows with the context; chunked prefill does not remove the head-of-line term.

3.6 Exercises

Exercise 3.1. Derive (3.2) from Theorem 3.4 and check cases (i) and (ii) of Corollary 3.5. *Hint:* $\mathbb{E}[S^2] = \mathbb{E}[S]^2(1 + \text{CV}^2)$ and $\lambda \mathbb{E}[S] = \rho$.

Exercise 3.2. Prove Corollary 3.7. *Hint:* $m_k(h)$ is affine in h with slope $\mathbb{E}[(S^{\text{hit}})^k] - \mathbb{E}[(S^{\text{miss}})^k] \leq 0$; in (ii) the numerator of W falls and its denominator rises.

Exercise 3.3. Prove Corollary 3.10. *Hint:* with $\rho = 1 - \varepsilon$ and $\varepsilon \leq 1/2$, the load term of (3.5) is $\lambda^2 m_2 \Delta S_i / (2\varepsilon^2) \geq m_2 \Delta S_i / (8m_1^2 \varepsilon^2)$.

Exercise 3.4. Prove Corollary 3.11. *Hint:* expand $P(K + n, 0) - P(n, K)$; each term of (3.5) is a nonnegative constant times ΔS , $(S^{\text{miss}} + S^{\text{hit}})\Delta S$ or ΔS , and $S^{\text{miss}} + S^{\text{hit}} = 2S^{\text{hit}} + \Delta S$.

Exercise 3.5. Show that the mean residual service time over busy time, $\mathbb{E}[S^2] / (2\mathbb{E}[S])$, is always at least $\mathbb{E}[S]/2$, with equality only when S is deterministic. Also check that it equals $\mathbb{E}[S]$ when S is exponential. *Hint:* $\mathbb{E}[S^2] = \mathbb{E}[S]^2 + \text{Var}[S]$.

Exercise 3.6. With the numbers of Example 3.9 and $q = 0.01$, compute ΔL_P exactly and check $0.1195 \leq \Delta L_P \leq 0.1252$. How does the ratio of the upper to the lower bound change as q grows? *Hint:* substitute $m_1 + \delta_1$ and $m_2 + \delta_2$ into $L_P = \lambda m_1 + \lambda^2 m_2 / (2(1 - \lambda m_1))$.

Chapter 4

Closed Systems and Memory: Finite Populations, MVA, Campbell's Theorem, Eviction

All models so far were **open systems**. Customers arrive from an infinite outside population, leave after service, and the arrival rate does not change when the queue is long. Agent sessions are different. A session submits a turn, receives the response, runs a tool, and only then submits its next turn. A waiting session cannot create a new turn, so the longer the queue, the fewer the arrivals. This chapter treats this *closed* structure (Section 4.1), computes the mean memory held by sessions (Section 4.2), and solves the combinatorial optimisation problem of what to drop when memory runs short (Section 4.3).

4.1 Closed systems and mean value analysis

The simplest closed system (Section 1.1) is the finite-source model $M/M/1//N$ (Definition 1.4(f)). N customers (sessions) alternate between a delay station, where each thinks for an exponential time of mean Z , and a FIFO server with service times $\text{Exp}(\mu)$: a session submits a turn when it finishes thinking and starts thinking again when the turn is served.

In LLM serving

Sessions are the customers of the closed loop. The think time is the time for tool execution and user response, and the server is the prefill stage. A session waiting for prefill cannot submit its next turn. A KV memory pool holds a few million tokens and real agent contexts reach hundreds of thousands of tokens, so a replica holds about 10 live sessions at a time. At this scale the $N = \infty$ approximation can be badly wrong.

Let $X(t)$ be the number of turns at the server (waiting or in service) at time t . There are $N - X(t)$ thinking sessions, each finishing its think time independently at rate $1/Z$ (Proposition 1.6), so X is a birth–death process on $\{0, 1, \dots, N\}$ with birth rates $\lambda_n = (N - n)/Z$ and death rates $\mu_n = \mu$ ($n \geq 1$). By Theorem 2.6 the stationary law is

$$\pi_N(n) = \frac{1}{G_N} \prod_{k=0}^{n-1} \frac{(N - k)/Z}{\mu} = \frac{1}{G_N} \frac{N!}{(N - n)!} \left(\frac{1}{\mu Z} \right)^n, \quad n = 0, 1, \dots, N, \quad (4.1)$$

with normalising constant G_N . The state space is finite, so no stability condition is needed.

In an open system PASTA (Theorem 1.14) told us that an arriving customer sees the time-average law. In a closed system arrivals are not Poisson, and an arriving customer does not see the time-average law. Instead the following holds.

Theorem 4.1 (Arrival theorem [21, 14])

In M/M/1//N in steady state, the law of the number of turns at the server seen by an arriving turn (excluding itself), that is, the long-run fraction $a_N(n)$ of arrivals that occur in state n , equals the time-stationary law of the same system with $N - 1$ sessions: $a_N(n) = \pi_{N-1}(n)$, $n = 0, \dots, N - 1$.

Proof. In state n arrivals occur at rate $(N - n)/Z$. By the ergodic theorem (Theorem 2.4) and the law of large numbers for jump counts, the number of arrivals in state n during $[0, t]$, divided by t , converges almost surely to $\pi_N(n)(N - n)/Z$. Hence

$$a_N(n) = \frac{(N - n) \pi_N(n)}{\sum_{m=0}^N (N - m) \pi_N(m)}.$$

By (4.1), for $n \leq N - 1$ we have $(N - n) \frac{N!}{(N - n)!} = N \frac{(N - 1)!}{(N - 1 - n)!}$, and for $n = N$, $(N - n) \pi_N(n) = 0$. Therefore $a_N(n) \propto \frac{(N - 1)!}{(N - 1 - n)!} (\mu Z)^{-n} \propto \pi_{N-1}(n)$, and since both are probability distributions they are equal. $\square$

Remark 4.2 (Back to PASTA as $N \rightarrow \infty$)

Fix $\lambda > 0$ and $\mu > \lambda$, and let the think time grow with the population, $Z = N/\lambda$, so that the total arrival rate when all sessions think stays λ . Then π_N and π_{N-1} both converge to the M/M/1 law $(1 - \rho)\rho^n$ (Exercise 4.1), so in the limit an arriving turn sees the time-average law: this is PASTA. For finite N the arrival theorem measures how far the closed system is from that limit.

The arrival theorem lets us solve a closed system by recursion on the number of sessions. With n sessions, let Q_n be the stationary mean number of turns at the server, R_n the mean response time of a turn (wait plus service), and X_n the throughput (turns completed per unit time).

Theorem 4.3 (Mean value analysis [21])

In M/M/1//n, $Q_0 = 0$ and for $n \geq 1$

$$R_n = \frac{1 + Q_{n-1}}{\mu}, \quad X_n = \frac{n}{Z + R_n}, \quad Q_n = X_n R_n.$$

Hence $Q_n = n(1 + Q_{n-1})/(\mu Z + 1 + Q_{n-1})$, the server utilisation is $\rho_n = X_n/\mu = n/(\mu Z + 1 + Q_{n-1})$, and the mean waiting time of a turn (excluding service) is $W_n = Q_{n-1}/\mu$.

Proof. By Theorem 4.1 an arriving turn sees on average Q_{n-1} turns ahead of it. Under FIFO all of them must finish before its service begins. By memorylessness (Proposition 1.6) the remaining time of the turn in service again has mean $1/\mu$, and each waiting turn has mean $1/\mu$, so the mean wait is Q_{n-1}/μ , and the response time is this plus the turn's own service $1/\mu$, which is R_n . One cycle of a session takes $Z + R_n$ on average. Applying Little's law (Theorem 1.17) to the whole system (all n sessions, always n customers) gives $n = X_n(Z + R_n)$, and applying it to the server gives $Q_n = X_n R_n$. The utilisation is $\rho_n = X_n/\mu$ by Corollary 1.18, and substitution gives the remaining formulas. $\square$

Proposition 4.4 (Finite population)

Let $\gamma = \mu Z > 0$ and define Q_n by the recursion of Theorem 4.3.

- (i) Q_n is nondecreasing in n , $0 \leq Q_n \leq n$, and Q_n is nonincreasing in γ . Moreover $\rho_N < 1$ for all $N \geq 1$.
- (ii) $W_N \leq \rho_N/(\mu(1 - \rho_N))$. The right side is the open M/M/1 waiting time at the same utilisation (Proposition 2.7).

Proof. Put $g_\gamma(q) = (1 + q)/(\gamma + 1 + q)$, so that $Q_n = n g_\gamma(Q_{n-1})$. On $q \geq 0$, g_γ is nondecreasing in q , nonincreasing in γ , and $0 < g_\gamma \leq 1$.

(i) $0 \leq Q_n \leq n$ follows by induction from $g_\gamma \leq 1$. $Q_1 = g_\gamma(0) \geq 0 = Q_0$, and if $Q_n \geq Q_{n-1}$ then $Q_{n+1} = (n+1)g_\gamma(Q_n) \geq n g_\gamma(Q_{n-1}) = Q_n$. For $\gamma \leq \gamma'$, assuming $Q_{n-1}(\gamma') \leq Q_{n-1}(\gamma)$, $Q_n(\gamma') = n g_{\gamma'}(Q_{n-1}(\gamma')) \leq n g_{\gamma'}(Q_{n-1}(\gamma)) \leq n g_\gamma(Q_{n-1}(\gamma)) = Q_n(\gamma)$. $\rho_N < 1$ is equivalent to $Q_{N-1} > N-1-\gamma$, so it suffices to show $Q_n > n-\gamma$ for all $n \geq 0$ by induction. For $n=0$ this is $0 > -\gamma$, and a computation gives

$$Q_{n+1} - (n+1-\gamma) = \frac{\gamma(Q_n + \gamma - n)}{\gamma + 1 + Q_n} > 0.$$

(ii) Since $\rho_N < 1$, the inequality is equivalent to $Q_{N-1}(1-\rho_N) \leq \rho_N$, that is, $Q_{N-1} \leq \rho_N(1+Q_{N-1}) = Q_N$, which is (i). $\square$

Example 4.5

With $N=2$ and $Z=2/\mu$ ($\gamma=2$), $Q_1=1/3$, $\rho_2=2/(3+1/3)=3/5$ and $W_2=1/(3\mu)$. The open formula at the same utilisation gives $\rho_2/(\mu(1-\rho_2))=3/(2\mu)$, which is 4.5 times the actual wait.

Remark 4.6

For exponential work and at the same utilisation, the open formula (Theorem 3.4) overestimates the wait (Proposition 4.4(ii)). For general work, and for the price of a miss built on the open formula (Theorem 3.8), this comparison has not been proved. Moreover, at most N turns exist, so $0 \leq L_P \leq N$, and no policy change can raise the mean number of turns at the server by more than $N - L_P$. This cap does not depend on the work law.

Remark 4.7

We assumed exponential think times, but the results depend only on their mean Z . The delay station is insensitive like the PS station (Theorem 2.14), and by the BCMP theorem a network of delay stations and PS stations has a product-form stationary law [3]. The service times of a FIFO server do not generalise in this way.

4.2 M/G/ ∞ and Campbell's theorem

We now count memory. Live sessions do not wait for each other and each keeps its own KV cache, so from the point of view of memory the sessions are customers of a delay station.

The model is the M/G/ ∞ system (Definition 1.4(e)): a delay station fed by Poisson arrivals of rate λ , with i.i.d. service times S independent of the arrivals.

The following lemma is a standard property of the Poisson process.

Lemma 4.8 (Conditional uniformity [22, §2.3])

For a Poisson process of rate λ , the number of arrivals in an interval $(a, b]$ is $\text{Poisson}(\lambda(b-a))$, and conditional on this number being n , the arrival times (as an unordered set) have the law of n i.i.d. samples from the uniform distribution on $(a, b]$.

Proposition 4.9 (M/G/ ∞ occupancy [22])

If $\mathbb{E}[S] < \infty$, then in an M/G/ ∞ system started empty at time $-t$, the number of customers at time 0 is $\text{Poisson}(\lambda \int_0^t \mathbb{P}(S > u) du)$, and as $t \rightarrow \infty$ it converges in distribution to $\text{Poisson}(\lambda \mathbb{E}[S])$.

Proof. Let A be the number of arrivals in $(-t, 0]$. Conditional on $A = n$, the arrival times are i.i.d. uniform (Lemma 4.8) and the service times are independent of them. A customer who arrived at time $-u$ is still present at time 0 only if $S > u$, so each customer remains independently with probability

$p_t = \frac{1}{t} \int_0^t \mathbb{P}(S > u) du$. Hence the number X remaining satisfies

$$\mathbb{P}(X = k) = \sum_{n \geq k} e^{-\lambda t} \frac{(\lambda t)^n}{n!} \binom{n}{k} p_t^k (1 - p_t)^{n-k} = e^{-\lambda t p_t} \frac{(\lambda t p_t)^k}{k!}.$$

Finally $\lambda t p_t = \lambda \int_0^t \mathbb{P}(S > u) du \rightarrow \lambda \int_0^\infty \mathbb{P}(S > u) du = \lambda \mathbb{E}[S]$. $\square$

That the limit depends on the law of S only through its mean is the insensitivity of the delay station. When the quantity each customer holds (for example, KV bytes) changes over time, the next theorem gives its mean.

Theorem 4.10 (Campbell's mean formula [11])

Let $\{T_k\}$ be the points of a Poisson process of rate λ on $\mathbb{R}$, and attach to each point an i.i.d. mark $Y_k \sim Y$ (with values in an arbitrary measurable space), independent of the points. For measurable $f \geq 0$,

$$\mathbb{E} \left[\sum_k f(T_k, Y_k) \right] = \lambda \int_{\mathbb{R}} \mathbb{E}[f(t, Y)] dt.$$

Proof. First consider the sum over the points in a bounded interval $I = (a, b]$ only. Conditional on the number of points being n , the pairs (T_k, Y_k) are n i.i.d. copies of (U, Y) with $U \sim \text{Uniform}(I)$ and Y independent (Lemma 4.8). Hence

$$\mathbb{E} \left[\sum_{T_k \in I} f(T_k, Y_k) \right] = \sum_{n \geq 0} \mathbb{P}(A_I = n) n \mathbb{E}[f(U, Y)] = \lambda(b - a) \cdot \frac{1}{b - a} \int_a^b \mathbb{E}[f(t, Y)] dt.$$

(The second equality uses Tonelli's theorem.) For $I_m = (-m, m]$ both sides are nondecreasing in m , and the monotone convergence theorem gives the claim as $m \rightarrow \infty$. $\square$

Corollary 4.11 (Mean resident KV)

Let sessions arrive in a Poisson process of rate Λ , and let session k have an i.i.d. lifetime $T_s^{(k)}$ and a context of $K_k(u) \geq 0$ tokens at age u (the lifetime and the trajectory may be dependent). In steady state (started at time $-\infty$), the mean number of resident KV tokens at a given instant is

$$\Lambda \mathbb{E} \left[\int_0^{T_s} K(u) du \right].$$

In particular, if $K(u) \equiv \bar{K}$ is constant for each session, this equals $\Lambda(\mathbb{E}[T_s]\mathbb{E}[\bar{K}] + \text{Cov}(T_s, \bar{K}))$. It exceeds the naive product $\Lambda\mathbb{E}[T_s]\mathbb{E}[\bar{K}]$ if and only if $\text{Cov}(T_s, \bar{K}) > 0$, that is, when long sessions have large contexts.

Proof. Attach the mark $Y_k = (T_s^{(k)}, K_k(\cdot))$ and put $f(t, (T, K)) = K(-t)\mathbf{1}\{0 \leq -t < T\}$. The resident KV at time 0 is $\sum_k f(T_k, Y_k)$. By Theorem 4.10 and Tonelli, its mean is

$$\Lambda \int_{-\infty}^0 \mathbb{E}[K(-t)\mathbf{1}\{-t < T_s\}] dt = \Lambda \mathbb{E} \left[\int_0^{T_s} K(u) du \right].$$

If $K \equiv \bar{K}$, the integral is $T_s \bar{K}$, and $\mathbb{E}[T_s \bar{K}] = \mathbb{E}[T_s]\mathbb{E}[\bar{K}] + \text{Cov}(T_s, \bar{K})$. $\square$

Taking $K \equiv 1$ gives the mean number of sessions $\Lambda\mathbb{E}[T_s]$, in agreement with Little's law.

4.3 Eviction: the covering knapsack problem

When memory runs short, part of the KV of the paused programs must be dropped (**eviction**). Program i occupies $c_i > 0$ bytes, and dropping it has an expected cost $w_i \geq 0$ (for example, the resume probability p_i times the price of a miss, Theorem 3.8).

Definition 4.12 (Covering knapsack problem)

For a finite set I and a target $\Delta C > 0$, the **covering knapsack problem** is

$$\min_{S \subseteq I} w(S) := \sum_{i \in S} w_i \quad \text{s.t.} \quad c(S) := \sum_{i \in S} c_i \geq \Delta C.$$

Allowing $x \in [0, 1]^I$ in place of S and minimising $\sum w_i x_i$ subject to $\sum c_i x_i \geq \Delta C$ gives the **fractional version**, which assumes that cost is linear in the evicted fraction. We call $v_i = w_i/c_i$ the **price per byte**.

Tail blocks are not fractional items. Engines evict blocks from the tail of a cached prefix, not whole programs. Dropping the last d of K cached tokens adds

$$r_K(d) = P(n + d, K - d) - P(n, K) = a d + b(Kd - d^2/2), \quad 0 \leq d \leq K,$$

of prefill work. The marginal cost $a + b(K - d)$ falls as d grows, so the cost is concave in d , not linear. The threshold rule below is therefore exact for whole programs and for the linear relaxation, but it does not prove that greedy block eviction is optimal. Proposition 4.14 below also concerns whole programs.

Theorem 4.13 (Threshold rule [5])

- (i) For every $\theta \geq 0$, the set $T_\theta = \{i : v_i \leq \theta\}$ costs no more than any S with $c(S) \geq c(T_\theta)$.
- (ii) In the fractional version, the solution that sets $x_i = 1$ in increasing order of v_i and fills the last item partially to meet the target exactly is optimal (provided $c(I) \geq \Delta C$).

Proof. (i) On $S \setminus T_\theta$ we have $w_i > \theta c_i$, and on $T_\theta \setminus S$ we have $w_i \leq \theta c_i$, so

$$w(S) - w(T_\theta) = \sum_{S \setminus T_\theta} w_i - \sum_{T_\theta \setminus S} w_i \geq \theta(c(S \setminus T_\theta) - c(T_\theta \setminus S)) = \theta(c(S) - c(T_\theta)) \geq 0.$$

(ii) Let x^* be the greedy solution and θ the price per byte of the last (partial) item. Then $x_i^* = 1$ if $v_i < \theta$, $x_i^* = 0$ if $v_i > \theta$, and $\sum c_i x_i^* = \Delta C$. For any feasible x ,

$$\sum_i w_i(x_i - x_i^*) = \sum_i (w_i - \theta c_i)(x_i - x_i^*) + \theta \sum_i c_i(x_i - x_i^*).$$

Each term of the first sum is nonnegative (if $v_i < \theta$ both factors are ≤ 0 , if $v_i > \theta$ both are ≥ 0 , and if $v_i = \theta$ the term is 0). The second sum is $\theta(\sum c_i x_i - \Delta C) \geq 0$. $\square$

In the integer version, price-per-byte order can be arbitrarily bad. With target $\Delta C = 2$ and three items $(c, w) = (1, 1), (L, L), (2, 3)$, price-per-byte order (ties broken in favour of the earlier item) takes the first two items at cost $1 + L$, while the third item alone costs 3. The ratio $(1 + L)/3$ diverges as $L \rightarrow \infty$. The standard repair is the *guarded greedy* algorithm, which guesses the most expensive evicted item and fills the rest in price-per-byte order; it is known to be within a factor 2 of the optimum for all w_i [4].

Real systems evict by a key fixed in advance rather than by a price: shortest context first (shortest-first, SF), least recently used (LRU), time-to-live expiry (TTL), and so on. Such keys do not look at the resume probability p_i .

Proposition 4.14 (Price-blind keys)

Consider the covering knapsack problem with $w_i = p_i \Phi_i$.

- (i) Consider a rule that chooses one of two programs from any information except their resume probabilities. For every R there are resume probabilities $p \in (0, 1]$ under which either program alone meets the target and the program the rule chooses costs more than R times the other. Hence no such rule has a constant approximation ratio.
- (ii) If every program resumes ($p_i = 1$) and the price is only the quadratic term of the recompute work, so that $w_i = c_i^2$, then SF equals price-per-byte order, costs at most 2 times any feasible solution, and the constant 2 cannot be improved.

Proof. (i) Take two programs of the same size $c > 0$ that agree in everything the rule observes, and a target $\Delta C = c$. The rule's input does not depend on the resume probabilities, so it drops a fixed one, e ; call the other o . Set $\Phi_e = \Phi_o = \Phi > 0$, $p_e = 1$ and $p_o = 1/(|R| + 2)$. Then $w_e = \Phi > R\Phi/(|R| + 2) = R w_o$. The optimum is at most w_o , so the rule's ratio exceeds R .

(ii) Since $w_i/c_i = c_i$, price-per-byte order is SF. SF takes programs from the shortest until the target is met, so it returns a feasible solution whenever the pool holds the target. Suppose SF takes $t_1 \leq \dots \leq t_m$ in sorted order, and let $\ell = t_m$ and $\sigma = t_1 + \dots + t_{m-1} < \Delta C$ (it did not stop earlier). Let S be any feasible solution.

Step 1: $\ell^2 \leq w(S)$. Every program with $c_i < \ell$ is among the first $m - 1$, and these free only $\sigma < \Delta C$. So S contains a program with $c_i \geq \ell$.

Step 2. Put $f(x) = x(\ell - x)$ for $x < \ell$ and $f(x) = 0$ for $x \geq \ell$. Then $f \geq 0$ and $x^2 \geq \ell x - f(x)$ for all $x \geq 0$. Summing over S and using $c(S) \geq \Delta C$, and noting that the terms with $f(c_i) > 0$ occur only among $t_1, \dots, t_{m-1}$,

$$w(S) \geq \ell \Delta C - \sum_{j < m} t_j(\ell - t_j) = \ell(\Delta C - \sigma) + \sum_{j < m} t_j^2.$$

Step 3. Since $\Delta C - \sigma > 0$, $w(\text{SF}) = \sum_{j < m} t_j^2 + \ell^2 \leq w(S) - \ell(\Delta C - \sigma) + \ell^2 \leq w(S) + \ell^2 \leq 2w(S)$.

Optimality of the constant 2: with sizes $K, K + 1$ and $\Delta C = K + 1$, SF takes K and then $K + 1$ at cost $K^2 + (K + 1)^2$, while $\{K + 1\}$ costs $(K + 1)^2$. The ratio tends to 2 as $K \rightarrow \infty$. $\square$

Example 4.15

Even in the situation of (ii), SF is not optimal. With contexts 4, 5, 6 and target 6, SF drops $\{4, 5\}$ at cost $16 + 25 = 41$, while $\{6\}$ is feasible at cost 36.

Part (i) says that when resume probabilities differ between programs, no fixed key can stand in for the price. How large such differences are in practice is a question about the workload and must be measured.

4.4 Summary

- In a closed system the arrival theorem replaces PASTA, and mean value analysis gives the mean queue; for exponential work the open formula overstates the wait.
- Campbell's formula gives the mean resident KV, above the naive product when long sessions have long contexts.
- Eviction is a covering knapsack: the threshold rule solves its fractional version, and a key blind to resume probabilities has no constant ratio.

4.5 Exercises

Exercise 4.1. Prove the limit in Remark 4.2. *Hint:* with $Z = N/\lambda$, $\pi_N(n) \propto \frac{N!}{(N-n)!N^n}\rho^n$, and $\frac{N!}{(N-n)!N^n} = \prod_{k < n} (1 - k/N)$ lies in $[0, 1]$ and tends to 1; use dominated convergence with ρ^n .

Exercise 4.2. For $N = 2$ and $\gamma = \mu Z$, compute $\sum_n n \pi_2(n)$ directly from (4.1) and check that it equals Q_2 of Theorem 4.3. *Hint:* $G_2 = 1 + 2/\gamma + 2/\gamma^2$.

Exercise 4.3. Show that the stationary mean number of customers in $M/G/\infty$ is $\lambda \mathbb{E}[S]$ without using Proposition 4.9. *Hint:* at a delay station $W = \mathbb{E}[S]$, so use Little's law.

Exercise 4.4. Sessions arrive at rate Λ and live for T_s , with κ KV bytes per token. A session's context grows linearly, $K(u) = k_0 + gu$ at age u . Use Corollary 4.11 to show that the mean resident KV is $\kappa \Lambda (k_0 \mathbb{E}[T_s] + g \mathbb{E}[T_s^2]/2)$. For exponential T_s , compare it with the naive estimate $\kappa \Lambda \mathbb{E}[T_s] (k_0 + g \mathbb{E}[T_s]/2)$, which uses the mean lifetime alone. *Hint:* $\mathbb{E}[T_s^2] = 2\mathbb{E}[T_s]^2$ for an exponential lifetime.

Chapter 5

Colocated Serving: The Paper's Scheduler, Step by Step

The first four chapters built tools. This chapter assembles them into the argument of the paper. We go in the order the paper does: the model (Section 5.1), where the variance of prefill work comes from (Section 5.2), the decision problem and its single price (Section 5.3), and the scheduler whose every step compares that price with the price of memory (Section 5.4). We close with an honest list of what is exact and what is approximate (Section 5.5). For each step we name the result of Chapters 1–4 that carries it.

5.1 The model on one page

Replica and its three demands. The system has J **replicas**; each has a memory pool of M bytes. A turn i has context K_i tokens (already in the session), appends n_i new tokens and decodes o_i output tokens. It makes three demands on the replica that serves it.

(a) *Prefill compute.* Prefilling n tokens on top of a resident prefix of K tokens costs

$$P(n, K) = a n + b n (K + n/2) \quad (5.1)$$

seconds, with $a, b \geq 0$: a linear term for the dense layers and an attention term over the prefix. A **hit** costs $P(n_i, K_i)$; a **miss** recomputes the whole context, $P(K_i + n_i, 0)$.

(b) *Decode bandwidth.* $D_i = o_i(\beta K_i + \omega/m)$ seconds, where m is the batch size, β the time to read one token's KV and ω the time to read the weights once.

(c) *Memory.* $c_i = \kappa K_i$ bytes while the state is resident, with κ the KV bytes per token.

The hit/miss distinction lives entirely in demand (a): D_i and c_i are the same for a hit and a miss. By Corollary 3.11 the extra work of a miss is $\Delta S_i = a K_i + b K_i^2/2$, quadratic in the context.

Two stations and a pool. Figure 1.1 in Section 1.2 shows the structure; here we add the numbers and name the chapter that backs each modelling choice.

1. *Decode is a PS station* with capacity $\varphi(m)$ (Definition 2.9). Continuous batching advances every turn in the batch at each step. By Theorem 2.14 the batch-size law is $\pi(m) \propto \rho_D^m / \prod_{k \leq m} \varphi(k)$ with $\rho_D = \lambda \mathbb{E}[D]$, whatever the law of D .
2. *Prefill is a FIFO queue* whose server is available the fraction of time the decode batch leaves. Chunked prefill interleaves chunks with decode steps but, without a per-prefill cap, serves pending prefills in arrival order. With Poisson arrivals and a server of fixed speed, it is the M/G/1 queue of Chapter 3, with the PK wait $\mathbb{E}[W_q] = \lambda \mathbb{E}[S^2]/(2(1 - \rho))$ (Theorem 3.4).
3. *Sessions form a closed loop* (prefill $\rightarrow$ decode $\rightarrow$ tool), and new sessions arrive as a Poisson process of rate Λ . The tool call is a delay station (Section 1.1).

4. *The pool* of M bytes holds both the running batch and the paused sessions' KV. Once it is full, the KV policy decides what to drop, and so which future turns hit.

The objective. The paper minimises $L = L_P + L_D$, the mean number of turns in the prefill stage plus the mean number in the decode batch. By Little's law (Theorem 1.17), L equals λ times the mean time a turn spends in the two stages, so minimising L is minimising mean delay.

In LLM serving

The two stations respond to the cache in opposite ways. The decode stage sees only the mean demand (insensitivity), and a miss does not change the decode demand at all (Corollary 2.17). The prefill stage sees the second moment of its work (PK), and the cache sets that second moment. This asymmetry is the whole reason the paper prices misses at the prefill queue.

5.2 Where the variance of prefill work comes from

The PK formula makes $\mathbb{E}[S^2]$, equivalently CV^2 , a first-class quantity (Corollary 3.5). The paper does not assume that agentic workloads are intrinsically “high variance”; it asks where the variance comes from. The tool is the law of total variance.

Lemma 5.1 (Variance split of the hit/miss mixture)

Let S_h be the mixture (3.3): S^{hit} with probability h and S^{miss} with probability $1 - h$, with means $\mu_{\text{hit}}, \mu_{\text{miss}}$ and variances $\sigma_{\text{hit}}^2, \sigma_{\text{miss}}^2$. Then

$$\text{Var}[S_h] = \underbrace{h\sigma_{\text{hit}}^2 + (1-h)\sigma_{\text{miss}}^2}_{\text{spread of the appends}} + \underbrace{h(1-h)(\mu_{\text{miss}} - \mu_{\text{hit}})^2}_{\text{hit/miss mixture}}.$$

Proof. Let I be the indicator of a hit. By the law of total variance, $\text{Var}[S_h] = \mathbb{E}[\text{Var}(S_h | I)] + \text{Var}(\mathbb{E}[S_h | I])$. The first term is $h\sigma_{\text{hit}}^2 + (1-h)\sigma_{\text{miss}}^2$. The conditional mean $\mathbb{E}[S_h | I]$ takes the value μ_{hit} with probability h and μ_{miss} with probability $1 - h$, a two-point variable whose variance is $h(1-h)(\mu_{\text{miss}} - \mu_{\text{hit}})^2$. $\square$

The first term is not the scheduler's to change: it comes from the spread of tool outputs (a line or a whole file). The second term is the KV policy's lever. It vanishes at $h \in \{0, 1\}$ and grows with the squared gap $(\mu_{\text{miss}} - \mu_{\text{hit}})^2$, which by (5.1) grows like K^4 once attention dominates. The paper measures both terms on production traces rather than assuming either one dominates. The lesson for the theory is that what a scheduler must price is not a ratio of CV^2 but the rise of $\lambda \mathbb{E}[S^2]/(2(1-\rho))$ that one more miss causes. That is exactly Φ_i .

Remark 5.2 (Partial hits)

An engine may reuse only $C \leq K$ tokens of the prefix, for example because the cache keeps whole blocks only. The turn then costs $P(K+n-C, C)$, a full hit plus $r_K(K-C)$ of Section 4.3. A binary hit rate then no longer determines $\mathbb{E}[S]$ and $\mathbb{E}[S^2]$; one needs the distribution of C . Lemma 5.1 still holds for a two-point mixture, but its conditional means must be recomputed when partial hits are grouped with misses.

5.3 The decision problem and one price

At a decision epoch a replica holds a set of paused programs. Program i holds c_i bytes, resumes with probability p_i after an expected further τ_i seconds, and its next turn needs S_i^{hit} if its state is still resident and $S_i^{\text{hit}} + \Delta S_i$ if not. The controller decides for each state whether to keep it, and the memory constraint couples all these decisions.

Step 1: price a miss. Theorem 3.8 gives the rise in L_P when a fraction of turns turns from hits into misses:

$$\Phi_i = \underbrace{\Delta S_i}_{\text{own work}} + \underbrace{\frac{\lambda((S_i^{\text{miss}})^2 - (S_i^{\text{hit}})^2)}{2(1-\rho)}}_{\text{head-of-line}} + \underbrace{\frac{\lambda W \Delta S_i}{1-\rho}}_{\text{load}}.$$

A single eviction is a transient event, while Theorem 3.8 compares two stationary regimes. The paper therefore reads one eviction as a small change of policy, a fraction q_i of arrivals turning from hits into misses. This is an approximation, and the paper tests it empirically.

Step 2: price a drop. Dropping program i 's state causes a miss only if the program resumes, so the expected price of the drop, its price per byte and its price per byte-second held are

$$w_i = p_i \Phi_i, \quad v_i = \frac{w_i}{c_i}, \quad u_i = \frac{w_i}{c_i \tau_i}. \quad (5.2)$$

Step 3: price memory. Keeping program i occupies c_i bytes for about τ_i seconds, that is $c_i \tau_i$ byte-seconds. Let $x_i \in \{0, 1\}$ indicate “keep” and let $\mathcal{B}$ be the byte-seconds available over the horizon. The keep-or-drop problem is

$$\text{OPT} = \min_{x \in \{0,1\}^I} \sum_i (1 - x_i) w_i \quad \text{s.t.} \quad \sum_i c_i \tau_i x_i \leq \mathcal{B}. \quad (5.3)$$

Relaxing the constraint with a multiplier $\theta \geq 0$ gives the Lagrangian $\mathcal{L}(x, \theta) = \sum_i (1 - x_i) w_i + \theta (\sum_i c_i \tau_i x_i - \mathcal{B})$. We call θ the **shadow price** of a byte-second.

Proposition 5.3 (One price for memory)

- (i) For every $\theta \geq 0$, $x_i^\theta = \mathbf{1}\{u_i > \theta\}$ minimises $\mathcal{L}(\cdot, \theta)$, and $g(\theta) := \mathcal{L}(x^\theta, \theta) \leq \text{OPT}$.
- (ii) If θ is such that x^θ uses the budget exactly, $\sum_i c_i \tau_i x_i^\theta = \mathcal{B}$, then x^θ is optimal for (5.3).

Proof. (i) $\mathcal{L}(x, \theta) = \sum_i w_i - \theta \mathcal{B} + \sum_i x_i (\theta c_i \tau_i - w_i)$ is separable in x . Each term is minimised by $x_i = 1$ when $\theta c_i \tau_i < w_i$, that is $u_i > \theta$, and by $x_i = 0$ otherwise. For any feasible x the bracket $\sum_i c_i \tau_i x_i - \mathcal{B}$ is ≤ 0 , so $\sum_i (1 - x_i) w_i \geq \mathcal{L}(x, \theta) \geq g(\theta)$. Minimising over feasible x gives $\text{OPT} \geq g(\theta)$ (weak duality).

(ii) x^θ is feasible, and its constraint term vanishes, so its objective equals $g(\theta) \leq \text{OPT}$. Hence it is optimal. $\square$

Part (ii) is the threshold rule of Theorem 4.13 in byte-second form: keep what is worth more per byte-second than memory, drop the rest. When no θ uses the budget exactly, because items are indivisible, the threshold rule with one item split still solves the fractional relaxation. Tail blocks do not fit this relaxation (Section 4.3).

Step 4: how long to keep. Proposition 5.3 decides at one instant. Because the Lagrangian is separable, once θ is fixed each program can also be treated on its own over time: keep its state until some time T and drop it then if the program has not resumed. Let the program resume with probability $p \in (0, 1]$, and, given that it resumes, after a time with a continuous distribution function G and density g on $[0, \infty)$. Holding the state costs θc per second; dropping it costs Φ if the program resumes later. The expected cost of the rule “hold until T ” is

$$C(T) = \theta c \int_0^T (1 - pG(t)) dt + \Phi p (1 - G(T)), \quad (5.4)$$

since the state is held at time t exactly when the program has not resumed by t , and a miss occurs exactly when it resumes after T . The quantity that decides is the **resume hazard**

$$\eta(t) = \frac{p g(t)}{1 - pG(t)},$$

the rate at which a program that has not resumed by t resumes at t .

Proposition 5.4 (Optimal holding time)

Let $c > 0$, $\theta > 0$ and $r = \theta c / \Phi$ with $\Phi > 0$. Suppose g is continuous. Define $\eta(T) = 0$ after $pG(T) = 1$.

- (i) $C'(T) = (1 - pG(T))(\theta c - \Phi \eta(T))$ for every $T \geq 0$. Holding a little longer lowers the cost exactly when $\Phi \eta(T) > \theta c$.
- (ii) If η is nonincreasing, let $T^* = \inf\{T \geq 0 : \eta(T) \leq r\}$. If $T^* < \infty$ it minimises C over $[0, \infty)$; if $T^* = \infty$, C is nonincreasing and the state should be held until the program resumes. In particular $T^* = 0$, drop at once, if and only if $\eta(0) \leq r$.
- (iii) If G is exponential with mean τ , then η is nonincreasing and $\eta(0) = p/\tau$, so the state should be kept at all if and only if

$$u = \frac{p\Phi}{c\tau} > \theta,$$

the price per byte-second of (5.2). If $p = 1$ and $u > \theta$, the state is held until the program resumes. If $p < 1$ and $u > \theta$, the optimal holding time is

$$T^* = \tau \ln \left[\frac{p}{1-p} \left(\frac{\Phi}{\theta c \tau} - 1 \right) \right].$$

Proof. (i) The integrand in (5.4) is continuous, so by the fundamental theorem of calculus the first term has derivative $\theta c(1 - pG(T))$, and the second has derivative $-\Phi p g(T)$. Factoring out $1 - pG(T)$ gives the formula when $pG(T) < 1$; when $pG(T) = 1$ (possible only if $p = 1$) both sides vanish, because $g(T) = 0$ wherever G has reached 1 and g is continuous. The factor $1 - pG(T)$ is nonnegative, so the sign of $C'(T)$ is the sign of $\theta c - \Phi \eta(T)$.

(ii) For $T < T^*$, $\eta(T) > r$, so $C'(T) \leq 0$ by (i). For $T > T^*$, $\eta(T) \leq r$ because η is nonincreasing, so $C'(T) \geq 0$. Hence C is nonincreasing on $[0, T^*]$ and nondecreasing on $[T^*, \infty)$, and T^* is a minimiser. If $T^* = \infty$, then $C' \leq 0$ everywhere. The last claim is the definition of T^* at $T = 0$, using that η is nonincreasing.

(iii) With $x = e^{-T/\tau}$, $g(T) = x/\tau$ and $1 - pG(T) = 1 - p + px$, so

$$\eta(T) = \frac{p}{\tau} \cdot \frac{x}{1 - p + px}.$$

The map $x \mapsto x/(1 - p + px)$ is nondecreasing on $(0, 1]$, and x decreases in T , so η is nonincreasing; it is the constant $1/\tau$ when $p = 1$. At $T = 0$, $\eta(0) = p/\tau$, and $\eta(0) > r$ is $p\Phi/(c\tau) > \theta$. If $p = 1$, $\eta \equiv 1/\tau > r$, so $T^* = \infty$ by (ii). If $p < 1$, $\eta(T) \rightarrow 0$ as $T \rightarrow \infty$, so T^* is finite and solves $\eta(T^*) = r$ by continuity: $px/\tau = r(1 - p + px)$, that is $x = r(1 - p)/(p(1/\tau - r))$, and $T^* = -\tau \ln x = \tau \ln \left[\frac{p}{1-p} \left(\frac{1}{r\tau} - 1 \right) \right]$ with $1/(r\tau) = \Phi/(\theta c \tau)$. $\square$

When $\theta = 0$, holding until resumption has zero cost and is optimal. For positive memory price, the instantaneous test $u_i > \theta$ of the scheduler is exactly the optimal decision to keep a state for exponential resume times, and part (iii) adds how long to keep it. For example, with $p = 0.5$, $\tau = 10$ s, $\Phi = 20$ s, $c = 1$ and $\theta = 0.3$, $u = 1 > \theta$, the optimal holding time is $T^* \approx 17.3$ s, and it lowers the expected cost from $C(0) = p\Phi = 10$ to $C(T^*) \approx 5.6$. When the gap distribution has a decreasing hazard, as heavy-tailed gaps typically do, part (ii) gives a program-specific time-to-live.

Remark 5.5 (Unknown resume times and the oracle)

If nothing is known about G , holding the state is the classical *ski rental* problem: pay rent θc per second or buy once at price Φ [7]. The rule “hold for $\Phi/(\theta c)$ seconds” costs at most twice the cost in hindsight for any resume time: if the program resumes at a time $s \leq \Phi/(\theta c)$ it pays θcs , which is optimal; otherwise it pays 2Φ against a hindsight optimum of Φ . The hindsight optimum, which knows whether and when each program resumes, is called the **oracle**; it keeps a state exactly when the program resumes and $\theta cs < \Phi$. It cannot be run online, but it is the benchmark against which a policy’s cost is reported as cost/OPT. The factor two needs the program to resume: against a program that never does, the oracle pays nothing, and no rule that holds the state for a positive time has a bounded ratio.

Proposition 5.4 holds θ and Φ fixed, while in the real system both move with the policy; the joint optimum is a Markov decision problem with no closed form.

5.4 The scheduler, step by step

Every decision now compares a program’s u_i with the replica’s θ .

Eviction. Freeing ΔC bytes now is the covering knapsack of Definition 4.12 with $w_i = p_i \Phi_i$. Price-per-byte order solves its fractional relaxation (Theorem 4.13); for whole programs the guarded greedy of Section 4.3 stays within a factor 2. Retention over time uses u_i instead. For tail blocks, whose cost is concave, neither order is proved optimal, and the scheduler compares the lost work r_K of each candidate tail. Deployed rules evict by a fixed key (shortest context, least recently used, longest idle, time-to-live). By Proposition 4.14 a key that ignores p_i has no constant approximation ratio, and shortest-first is within a factor two when every program resumes and the price is the quadratic recompute term alone.

Example 5.6 (The load reorders programs)

Take $a = 0.2$ ms/token and $b = a/K_c$ with $K_c = 50,000$ tokens, and an append of $n = 1000$ tokens. Program A has $K = 200$ k tokens and $p = 0.3$; program B has $K = 20$ k tokens and $p = 0.9$. Then $\Delta S_A = 120$ s and $\Delta S_B = 4.8$ s. At an idle queue ($\lambda \rightarrow 0$) the price is the own work, and the price per thousand tokens $p\Phi/K$ is 0.18 s for A and 0.216 s for B: evict A first. At $\lambda = 0.25$ /s with $\mathbb{E}[S] = 2$ s and $\mathbb{E}[S^2] = 20$ s² ($\rho = 0.5$, $W = 5$ s) the head-of-line term dominates; $\Phi_A \approx 4080$ s and $\Phi_B \approx 23.2$ s, and the prices per thousand tokens become 6.12 s and 1.05 s: evict B first. No fixed key reproduces both orders, which is why the scheduler prices at decision time.

Keep, offload or drop. With a slower tier (host memory or a shared KV store) each paused state has three options: keep it at cost $\theta c_i \tau_i$, drop it at expected cost w_i , or offload it at the **priced fetch**, which is Φ_i with the transfer time in place of ΔS_i (plus the tier’s own wait once it saturates). The scheduler takes the least of the three. Two facts follow from this being a minimum.

Proposition 5.7 (Offloading as an option)

Let each state i have a finite set of actions $\mathcal{A}_i$ with costs $\chi_i(a)$, and let the total cost be separable. Then

- (i) enlarging each $\mathcal{A}_i$ never raises the optimal cost; in particular, making offloading available never hurts;
- (ii) a policy that always offloads can cost more than one that never offloads.

Proof. (i) A minimum over a larger set is at most the minimum over a subset. (ii) One state with keep cost 1, drop cost 1 and priced fetch 2 suffices: always offloading pays 2, never offloading pays 1. $\square$

So when an always-offloading stack collapses, the collapse is a property of the policy, not of the offloading mechanism.

Placement. A resuming turn can run where its state is (replica 1) or elsewhere (replica j). The router charges each replica its wait W_j and service S_j , the cost R_j of moving or recomputing the state, and the expected future cost F_j of the program's remaining turns, and picks the minimum. The price of a miss supplies R_j : moving turns the next turn into a miss served at replica j , so R_j is Φ_i at replica j 's load. *Session affinity*, the rule that never moves a program, sets $R_j = \infty$ for $j \neq 1$.

Proposition 5.8 (When affinity loses)

Let the affinity replica be an M/M/1 queue with service rate μ and utilisation $\rho < 1$, and let the alternative be an idle replica with the same rate, at extra cost $x = R + F \geq 0$. Staying costs more than moving if and only if

$$\rho > \rho^* := \frac{\mu x}{1 + \mu x},$$

and ρ^* is nondecreasing in x .

Proof. See Exercise 5.2. □

An idle alternative gives the earliest inversion, so ρ^* is a lower bound on the load at which affinity loses to any real alternative. A shared KV store lowers R and hence ρ^* : affinity and a store are alternatives, not complements.

Admission. The batch and the paused states compete for one pool, and unstably: a miss keeps its turn longer in prefill and in the batch, which holds more bytes, evicts more paused state and causes more misses. Admission caps the number of live sessions to stay out of this regime. By Corollary 4.11 and Little's law the mean resident KV is (live sessions) $\times \bar{k}$, with $\bar{k} = \kappa \mathbb{E}[\int_0^{T_s} K(t) dt] / \mathbb{E}[T_s]$ the time-averaged footprint of a session, which exceeds the naive $\kappa \mathbb{E}[\bar{K}]$ when long contexts belong to long sessions. The cap is near $M/\bar{k}$; a session is worth admitting while its memory, priced at θ per byte-second, costs less than the wait at the entry queue.

The scheduler in one table. Each deployed rule replaces one price by a constant:

Rule	What it fixes
shortest-first eviction	$w_i \propto c_i^2$ (every program resumes, quadratic term only)
LRU, time-to-live	p_i constant
always offload	priced fetch = 0
strict session affinity	$R_j = \infty$
admit every session	$\theta = 0$

5.5 What is exact and what is approximate

A mathematician should know which statements are theorems about the model and which are modelling choices.

- *Exact, given the model:* the price bracket (Theorem 3.8), the decode price and its monotonicity (Theorem 2.12, Proposition 2.16), the finite-population inequalities (Proposition 4.4), the price-blind bounds (Proposition 4.14), and Propositions 5.3, 5.7 and 5.8.
- *Approximate:* the prefill stage is not M/G/1. Session feedback makes its arrivals non-Poisson, and the server's budget fluctuates with the decode batch. BCMP product form covers the PS and delay stations but not the FIFO prefill stage. For exponential work, Proposition 4.4(ii) shows the open formula overstates the wait at equal utilisation. For general work this comparison is an empirical question.

- *Approximate*: one eviction is a transient event priced by a stationary comparison (Step 1 above).
- *A modelling choice*: φ is fixed by a step rule that serves the decode batch at every step. An engine that gives a prefill a step of its own stops its decodes for that step, so its decode capacity falls with the prefill load; there a miss adds no decode demand but shrinks the capacity, and a share $f \in (0, 1]$ of the capacity at the same demand is the load ρ_D/f , so a smaller share leaves more turns in the batch (Theorem 2.12). Proposition 2.16 and Corollary 2.17 are about the first kind of engine.
- *Not in the queueing formula*: a turn whose KV does not fit waits for memory blocks before its prefill starts. That wait is set by the pool, not by the queue, and adds to the PK wait. The capacity function φ depends on the footprint law and the admission rule and must be calibrated, not derived.

What the testbed showed about the last point. On the paper’s testbed with contexts of about 50k tokens, the wait for the first token was a wait for memory blocks. The prefill server was busy only a small fraction of the time, and the PK wait was one to two orders of magnitude below the observed wait. Three engine rules made the pool a queue: a turn is admitted only when its whole prompt fits, admission is in arrival order, and each admission takes the least recently used cached blocks. The turns that missed had the same think times as the hits, and nearly all of them had arrived while their engine instance already had a queue. So the misses were selected by congestion, and on that testbed the ratio of miss to hit waits measured congestion, not the price of a miss. In the short-context runs, the extra misses after forced ones followed long think gaps and frequent re-insertions of contexts, which points to the pool rather than to the wait. In other words, the hit rate is not a policy input: it depends on the wait, and the wait depends on the hit rate. Section 6.7 models this feedback.

The paper tests the approximate links in two ways: in simulation on replayed production sessions, and on a real testbed. The criterion is *decision faithfulness*: the model must rank policies the way the real system does, even where its latencies are off.

5.6 Summary

- Prefill sees the second moment of its work and decode only the mean, so a miss is priced at the prefill queue.
- One shadow price θ turns eviction, holding time, offloading, placement and admission into comparisons with $u_i = p_i \Phi_i / (c_i \tau_i)$.
- Each deployed rule fixes one of these prices to a constant.

5.7 Exercises

Exercise 5.1. In Lemma 5.1 let $\sigma_{\text{hit}} = \sigma_{\text{miss}} = \sigma$. Find the hit rate h that maximises the mixture term, and show that the mixture term exceeds the append term exactly when $h(1-h)(\mu_{\text{miss}} - \mu_{\text{hit}})^2 > \sigma^2$. *Hint*: $h(1-h)$ is maximised at $h = 1/2$.

Exercise 5.2. Prove Proposition 5.8 by comparing the M/M/1 response time $1/(\mu(1-\rho))$ with $1/\mu + x$. Then, with $\mu = 1$ turn/s, compute ρ^* of Proposition 5.8 for $x = 0.1, 1$ and 10 s, and interpret: how loaded must the affinity replica be before moving a program with a ten-second recompute pays off? *Hint*: $\rho^* = x/(1+x)$ when $\mu = 1$.

Exercise 5.3. In a colocated replica, prefill and decode share one instance and one pool, and each iteration’s token budget goes first to the decoding turns and the rest to the oldest prefill. A step stage in serQ (Appendix A) instead serves its jobs in their order of entry. Show that the two orders coincide

when turns enter in the order they are admitted and a prefill may take all the budget left: after every iteration every job in its decode phase precedes every job in its prefill phase. Give a two-job example showing that a cap on a single prefill's tokens per iteration breaks this. *Hint:* a prefill that does not finish in an iteration leaves no budget behind it.

Chapter 6

Disaggregated Serving

This chapter extends the tools of Chapters 1 to 4 and the colocated scheduler of Chapter 5 to a disaggregated deployment. There, prefill and decode run on separate instances with separate memory pools, and a link carries the KV cache between them. The chapter derives what a scheduler for it needs: what a miss costs at each place where it can happen, what each instance should keep in memory and for how long, how many sessions to admit, and when misses feed themselves.

6.1 The disaggregated replica

Figure 6.1 shows the model.

Workload and costs. Sessions arrive as a Poisson process of rate Λ (Definition 1.8). Turn j of session i has a cached prefix of K_{ij} tokens, n_{ij} new tokens and o_{ij} output tokens, with context $\ell_{ij} = K_{ij} + n_{ij}$ and

$$K_{i1} = 0, \quad K_{i,j+1} = \ell_{ij}, \quad n_{i,j+1} \geq o_{ij}, \quad (6.1)$$

since the next turn's new tokens contain the previous output and the tool's output. After each turn the session calls a tool for a time with mean Z . A replica has prefill constants $a, b > 0$, κ KV bytes per token, a link of bandwidth B bytes per second with overhead x_0 , a decode capacity φ , and pools of M_P and M_D bytes. With the hit indicator H_{ij} (the prefix is still cached at admission), turn (i, j) has

$$S_{ij} = P(n_{ij}, K_{ij}) + (1 - H_{ij}) \Delta S(K_{ij}), \quad X_{ij} = x_0 + \frac{\kappa \ell_{ij}}{B}, \quad D_{ij} = o_{ij}, \quad c_{ij} = \kappa \ell_{ij}, \quad (6.2)$$

the prefill work in seconds (P and $\Delta S(K) = aK + bK^2/2$ as in Section 5.1 and Corollary 3.11), the link work in seconds, the decode work in tokens, and the footprint in bytes on either instance. A batch of m decodes takes $t(m)$ seconds per step, so $\varphi(m) = m/t(m)$ tokens per second.

Path of a turn.

- (i) A turn waits in arrival order at the prefill pool until $c = \kappa \ell$ bytes of running turns' allocations are free. Cached bytes are evicted as needed and never block admission. At admission the turn uses its own cached prefix if it survived; a hit is all or nothing.
- (ii) It receives S seconds of FIFO prefill, which produces the first token, and X seconds of link work. It then releases its prefill allocation and leaves its $\kappa \ell$ bytes cached. Ending a session does not erase this prefix.
- (iii) It waits in the same way for c bytes at the decode pool, receives D tokens of processor-sharing decode, and releases the allocation.
- (iv) With probability $p < 1$ the session calls a tool and submits another turn; otherwise it ends. So the number of turns J has $\mathbb{E}[J] = 1/(1 - p)$.

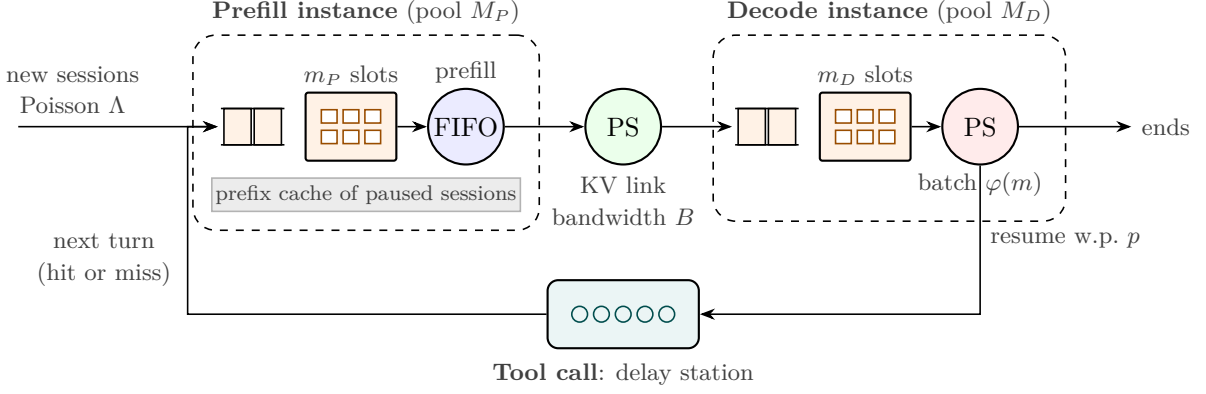


Figure 6.1: A disaggregated deployment as a queueing model. A turn waits for KV memory at a prefill instance (it is admitted only when its whole context fits, in arrival order), is prefilled one request at a time in arrival order, sends its KV cache over a link whose bandwidth concurrent transfers share, waits for KV memory at a decode instance, and decodes in a batch that shares the instance’s throughput. Sessions loop through a tool call back to the prefill instance, where the prefix cache decides whether the next turn is a hit (short prefill work) or a miss (the whole context is recomputed).

A FIFO stage serves only its first turn, at rate one; a PS stage with m turns serves each at rate $\varphi(m)/m$; the tool serves at rate one without waiting. Allocated plus cached bytes never exceed a pool.

The executable transfer of Appendix A reserves the decode pool before the transfer, which changes how long each pool is held.

6.2 Sample paths

Number the turns in order of arrival at the prefill pool, and let turn k of session $\sigma(k)$ have the epochs $a_k \leq \alpha_k \leq s_k \leq f_k \leq e_k \leq \alpha'_k \leq d_k$: arrival, admission, start and end of prefill, end of transfer, admission at decode, and end of decode. Let $R_P(t) = \{k : \alpha_k \leq t < e_k\}$ be the turns holding prefill memory. Then (Exercise 6.1)

$$\alpha_k = \inf \left\{ t \geq \max(a_k, \alpha_{k-1}) : \sum_{l < k, l \in R_P(t)} c_l + c_k \leq M_P \right\}, \quad (6.3)$$

$$s_k = \max(\alpha_k, f_{k-1}), \quad f_k = s_k + S_k, \quad (6.4)$$

$$a_{k^+} = d_k + Z_k \quad \text{if the session continues,} \quad (6.5)$$

where k^+ is the next turn of $\sigma(k)$, and $\alpha_0 = f_0 = 0$ for a path that starts empty at time 0. Eq. (6.4) is the FIFO recursion of Chapter 3, and (6.5) is the closed loop. If the system is stable, the turn rate is still fixed by the session rate, $\lambda = \Lambda \mathbb{E}[J] = \Lambda/(1-p)$. The link and decode batch are PS stations, the tool a delay station. If every context needs about c bytes, the equal-footprint approximation of Section 2.3, each pool is a slot queue with $m_j = \lfloor M_j/c \rfloor$ slots; a prefill slot is held through the transfer, until e_k .

The cache chooses the prefill work. The eviction order decides H_k , so with hit rate $h = \mathbb{E}[H]$

$$\mathbb{E}[S] = \mathbb{E}[P(n, K)] + \mathbb{E}[(1-H) \Delta S(K)], \quad (6.6)$$

and only the second term depends on the policy. Lemma 5.1 splits the variance of this work.

Performance. The first token appears when the prefill ends, so

$$\text{TTFT}_k = f_k - a_k = \underbrace{(\alpha_k - a_k)}_{W_{P,k}^M} + \underbrace{(s_k - \alpha_k)}_{W_{P,k}} + S_k, \quad (6.7)$$

the wait for prefill memory, the wait for the prefill server (W_P is its mean), and the prefill work. Transfer and decode reach the TTFT only through memory, because a prefill slot is held until e_k . A client that sees its first token only after the handoff also waits for the link and the decode pool, so its TTFT is $\alpha'_k - a_k$; the executable example of Appendix A records both, and the transfer apart, so that a comparison of colocated and disaggregated serving uses the same first-token basis. No stage is stable unless

$$\rho_P = \lambda \mathbb{E}[S] < 1, \quad \rho_L = \lambda \mathbb{E}[X] < 1, \quad \lambda \mathbb{E}[D] < \max_{m \leq m_D} \varphi(m), \quad (6.8)$$

and by (6.6) the first condition already depends on the eviction policy.

6.3 Three prices

A decision that changes where a paused session's KV lives changes the work of its next turn at one of three stations.

(a) A miss at the prefill instance. Under independent Poisson arrivals and i.i.d. prefill work, the prefill instance is an M/G/1 FIFO queue, so Theorem 3.8 applies as it stands: turning a fraction q_i of arrivals from hits into misses raises the mean number at the prefill instance by an amount in $[\lambda \sum_i q_i \Phi_i, \frac{1-\rho_P}{1-\rho'_P} \lambda \sum_i q_i \Phi_i]$, with

$$\Phi_i = \Delta S_i + \frac{\lambda((S_i^{\text{miss}})^2 - (S_i^{\text{hit}})^2)}{2(1 - \rho_P)} + \frac{\lambda W_P \Delta S_i}{1 - \rho_P}.$$

The middle term is the head-of-line term: every turn that arrives while the miss is prefilled waits for all of it.

(b) A miss at the decode instance. The path of Section 6.1 sends the whole context every turn. Suppose now that the decode instance also keeps the KV of paused sessions. If it still holds the session's KV from its previous turn, the next turn transfers only its new tokens, $\ell_i = n_i$ tokens; if not, it transfers the whole context, $\ell_i = K_i + n_i$. A decode-side miss therefore adds $\Delta X_i = \kappa K_i / B$ of link work.

Proposition 6.1 (The price of a transfer miss)

Let the link be a processor-sharing station of capacity 1 with Poisson arrivals of rate λ (an idealisation, since the link is fed by the prefill queue) and load $\rho_L < 1$, and turn a fraction q_i of arrivals into decode-side misses, raising the load to $\rho'_L = \rho_L + \lambda \sum_i q_i \Delta X_i < 1$. Then the mean number of transfers in progress rises exactly by

$$\Delta L_L = \frac{1 - \rho_L}{1 - \rho'_L} \lambda \sum_i q_i \Phi_i^X, \quad \Phi_i^X = \frac{\Delta X_i}{(1 - \rho_L)^2},$$

whatever the distribution of the transfer times.

Proof. By insensitivity (Theorem 2.14) the number of transfers in progress has the geometric law of Proposition 2.16 with capacity $C = 1$, for every transfer-time distribution with the given mean. Proposition 2.16 with $C = 1$ and ΔD_i replaced by ΔX_i gives the formula. $\square$

(c) Decode demand. A decision that adds decode demand, such as admitting a session, pays the load factor of Proposition 2.16 at the decode batch; a miss at either instance adds none (Corollary 2.17).

Comparison. The same context of K tokens costs the prefill instance a miss whose work is quadratic in K and whose price has a head-of-line term, set by the second moment of the prefill work; it costs the link a transfer linear in K whose price depends on its mean only, because under processor sharing a long transfer does not block the short ones. This growth comparison does not rank the pools universally: bandwidth, load, resume probability and holding time also set their memory values.

6.4 The decision problem with two pools

At a decision epoch each instance holds KV for a set of paused sessions. Session i holds c_i bytes, resumes with probability p_i after an expected further τ_i seconds, and its next turn pays Φ_i if its prefix is not in the prefill instance's cache and Φ_i^X if its KV is not at the decode instance. With indicators $x_i^P, x_i^D \in \{0, 1\}$ of “keep” at each instance and byte-second budgets $\mathcal{B}_P$ and $\mathcal{B}_D$, the keep-or-drop problem is

$$\text{OPT} = \min_{x^P, x^D} \sum_i [(1 - x_i^P) p_i \Phi_i + (1 - x_i^D) p_i \Phi_i^X] \quad \text{s.t.} \quad \sum_i c_i \tau_i x_i^P \leq \mathcal{B}_P, \quad \sum_i c_i \tau_i x_i^D \leq \mathcal{B}_D. \quad (6.9)$$

Relaxing the two constraints with multipliers $\theta_P, \theta_D \geq 0$, the **shadow prices** of a byte-second at each instance, gives the Lagrangian $\mathcal{L} = \sum_i [(1 - x_i^P) p_i \Phi_i + (1 - x_i^D) p_i \Phi_i^X] + \theta_P (\sum_i c_i \tau_i x_i^P - \mathcal{B}_P) + \theta_D (\sum_i c_i \tau_i x_i^D - \mathcal{B}_D)$.

Proposition 6.2 (One price per pool)

Let $u_i^P = p_i \Phi_i / (c_i \tau_i)$ and $u_i^X = p_i \Phi_i^X / (c_i \tau_i)$.

- (i) For all $\theta_P, \theta_D \geq 0$, $\mathcal{L}$ is minimised by keeping session i at the prefill instance iff $u_i^P > \theta_P$ and at the decode instance iff $u_i^X > \theta_D$, and the minimum $g(\theta_P, \theta_D)$ is at most OPT.
- (ii) If the multipliers are such that these choices use both budgets exactly, the choices are optimal for (6.9).

Proof. (i) $\mathcal{L} = \sum_i p_i (\Phi_i + \Phi_i^X) - \theta_P \mathcal{B}_P - \theta_D \mathcal{B}_D + \sum_i x_i^P (\theta_P c_i \tau_i - p_i \Phi_i) + \sum_i x_i^D (\theta_D c_i \tau_i - p_i \Phi_i^X)$ is separable: each x_i^P is best set to 1 exactly when $\theta_P c_i \tau_i < p_i \Phi_i$, that is $u_i^P > \theta_P$, and likewise each x_i^D . For a feasible (x^P, x^D) both brackets multiplying θ_P and θ_D are nonpositive, so its objective is at least its Lagrangian, which is at least g ; minimising over feasible choices gives $\text{OPT} \geq g$ (weak duality). (ii) The choices of (i) are then feasible and their constraint terms vanish, so their objective equals $g \leq \text{OPT}$. $\square$

This is the threshold rule of Theorem 4.13, once per pool: each instance keeps what is worth more per byte-second than its own memory.

Corollary 6.3 (Free transfer)

Suppose $\lambda x_0 < 1$. As the link bandwidth $B \rightarrow \infty$, $\Phi_i^X \rightarrow 0$ for every session, so for any $\theta_D > 0$ the decode instance keeps nothing for paused sessions, and all of M_D serves running decodes.

Proof. $\Delta X_i = \kappa K_i / B \rightarrow 0$, and $\rho_L \rightarrow \lambda x_0$ stays below 1, so $\Phi_i^X = \Delta X_i / (1 - \rho_L)^2 \rightarrow 0$ and $u_i^X \rightarrow 0 < \theta_D$. $\square$

A decode instance's cache is worth exactly the transfers it saves; the prefill instance's cache is worth the recomputation it saves and the queueing that recomputation would cause.

6.5 How long to keep

The holding-time proof in Proposition 5.4 applies separately to each pool. Use (Φ_i, θ_P) at prefill and (Φ_i^X, θ_D) at decode. The resume hazard, the exponential-gap holding time, and the unknown-gap

ski-rental comparison are unchanged. They hold the prices fixed; feedback changes those prices and is treated below.

6.6 How many sessions to admit

The admission rule of Section 5.4 applies with two changes. First, each instance has its own slot queue: with exponential holding times the wait for a slot is the Erlang C wait of Proposition 2.8, which grows without bound as the offered load of slots approaches m_P or m_D . Second, the Campbell estimate of resident KV (Corollary 4.11) is compared with the prefix cache of the prefill instance, and memory is priced at θ_P .

6.7 Misses that feed themselves

So far the hit rate was an input. It is also an output. A miss re-inserts a whole context into the prefill instance's pool, which evicts other paused prefixes sooner (*pool channel*); and a turn that waits longer is absent longer, which exposes its own prefix to eviction (*wait channel*). Let $W(h)$ be the mean wait at the prefill instance at hit rate h , nonincreasing in h (Corollary 3.7), and let $G(s; h)$ be the probability that a paused prefix survives an absence s when the instance runs at hit rate h , nonincreasing in s and nondecreasing in h .

Definition 6.4 (Feedback map)

The **feedback map** is $H(h) = \mathbb{E}_Z[G(Z + W(h); h)]$ on $[0, 1]$, and an **equilibrium** is a hit rate with $H(h) = h$.

The wait channel is an admission rule. Whether a waiting turn's prefix can be evicted is decided by the engine, not by the workload. An engine that takes the prefix out of the cache only when it schedules the turn (vLLM admits a waiting request at the start of an iteration with the budget left, and until then its cached blocks stay evictable) has the wait channel. An engine that pins the prefix when the turn arrives has survival $G(Z; h)$ instead of $G(Z + W(h); h)$, a larger value for every h , but it holds the pinned bytes and so lowers G for the other paused prefixes; the comparison of Proposition 6.5(ii) applies only when the first effect dominates. Pinning thus helps the pinned turn and hurts the other paused sessions. Which effect wins depends on the survival laws, so it must be computed for a given workload and memory size.

H is nondecreasing: if $h \leq h'$ then $W(h) \geq W(h')$, so $G(Z + W(h); h) \leq G(Z + W(h'); h) \leq G(Z + W(h'); h')$ for every Z , and taking expectations preserves the order. No continuity is needed for what follows.

Proposition 6.5 (Extremal equilibria (Knaster–Tarski) [25])

Let $H : [0, 1] \rightarrow [0, 1]$ be nondecreasing.

- (i) $\bar{h} = \sup\{x : x \leq H(x)\}$ is an equilibrium, and every equilibrium is at most $\bar{h}$; $\underline{h} = \inf\{x : H(x) \leq x\}$ is an equilibrium, and every equilibrium is at least $\underline{h}$.
- (ii) If $\tilde{H} : [0, 1] \rightarrow [0, 1]$ is nondecreasing and $H \leq \tilde{H}$ pointwise, then $\bar{h}_H \leq \bar{h}_{\tilde{H}}$ and $\underline{h}_H \leq \underline{h}_{\tilde{H}}$.

Proof. (i) Let $A = \{x \in [0, 1] : x \leq H(x)\}$; $0 \in A$, so $\bar{h} = \sup A$ exists. For $x \in A$, $x \leq H(x) \leq H(\bar{h})$ because $x \leq \bar{h}$ and H is nondecreasing; so $H(\bar{h})$ is an upper bound of A and $\bar{h} \leq H(\bar{h})$. Then $H(\bar{h}) \leq H(H(\bar{h}))$, so $H(\bar{h}) \in A$ and $H(\bar{h}) \leq \bar{h}$. Hence $H(\bar{h}) = \bar{h}$. Every equilibrium lies in A , so it is at most $\bar{h}$. The argument for $\underline{h}$ is the mirror image with $B = \{x : H(x) \leq x\} \ni 1$ and infima. (ii) $\bar{h}_H = H(\bar{h}_H) \leq \tilde{H}(\bar{h}_H)$, so $\bar{h}_H$ lies in the set A of $\tilde{H}$, and $\bar{h}_H \leq \bar{h}_{\tilde{H}}$. Likewise $H(\underline{h}_{\tilde{H}}) \leq \tilde{H}(\underline{h}_{\tilde{H}}) = \underline{h}_{\tilde{H}}$ puts $\underline{h}_{\tilde{H}}$ in the set B of H , so $\underline{h}_H \leq \underline{h}_{\tilde{H}}$. $\square$

A larger prefix cache raises G and so H , and by (ii) it raises both extremal equilibria. Other changes need care. In the open model the turn rate stays $\Lambda/(1-p)$, so a shorter think time or a faster decode acts through absence times and the number of live sessions; with a fixed population or an admission cap it also raises the turn rate. Either way the net effect on H has no sign in general.

Proposition 6.6 (Collapse)

In the open prefill instance take the wait of Theorem 3.4, $W(h) = \infty$ where $\lambda \mathbb{E}[S_h] \geq 1$, and $G(\infty; h) = 0$.

- (i) If $\lambda \mathbb{E}[S^{\text{miss}}] \geq 1$, then $H(0) = 0$, and every h at which the instance is overloaded is mapped to 0, which is mapped to itself: all-miss attracts the overloaded hit rates.
- (ii) If moreover $H(h_1) \geq h_1$ for some $h_1 > 0$, the least equilibrium is 0 and the greatest is at least h_1 : the instance has two distinct extremal equilibria.

Proof. (i) At $h = 0$ every turn misses, the load is $\lambda \mathbb{E}[S^{\text{miss}}] \geq 1$, so $W(0) = \infty$ and $H(0) = G(\infty; 0) = 0$. At an overloaded h , $H(h) = 0$ in the same way, and $H(0) = 0$. (ii) 0 is an equilibrium, so $\bar{h} \leq 0$; and $h_1 \in A$, so $\bar{h} \geq h_1$ by Proposition 6.5. $\square$

In a closed system the population bounds the wait and removes this mechanism for $H(0) = 0$, but not the pool channel: if re-inserted contexts evict every paused prefix within a think time, $H(0) = 0$ still. Thrashing, a collapse of this kind cured by limiting the load, is classical in paging systems; the admission cap of Section 6.6 is its cure here.

Example 6.7 (Two equilibria and a cap)

Take deterministic $S^{\text{hit}} = 0.5$ s, $S^{\text{miss}} = 5$ s, a deterministic think time $Z = 7$ s and the wait channel alone, $G(s) = e^{-s/(60\text{s})}$, with the Pollaczek–Khinchine wait. At $\lambda = 0.25/\text{s}$ the all-miss load is 1.25, $H = 0$ on $[0, 2/9]$, and a grid search finds three equilibria: 0 (stable), about 0.250 (unstable, slope $H' \approx 12$) and about 0.882 (stable, slope 0.07). At $\lambda = 0.18/\text{s}$ the all-miss load is 0.9 and the only equilibrium is about 0.885. Lowering the load removes the collapse without moving the good equilibrium much.

Proposition 6.8 (The forced-miss multiplier)

Let $H(h) = \nu + kh$ near an equilibrium, with $\nu \geq 0$ and $0 \leq k < 1$, so $h_0 = \nu/(1-k)$, and force a fraction $\delta \in [0, 1]$ of turns to miss, so the realised hit rate is $(1-\delta)H(h)$. The new equilibrium is $h_\delta = (1-\delta)\nu/(1-(1-\delta)k)$, and

$$h_0 - h_\delta = \frac{\delta \nu}{(1-k)(1-(1-\delta)k)} \geq \delta h_0.$$

Proof. See Exercise 6.4. $\square$

To first order in δ a forced miss converts a would-be hit with probability h_0 and induces $h_0 k/(1-k)$ further misses on other turns, each with its own price, so its marginal price is $h_0[\Phi_{\text{forced}} + \frac{k}{1-k}\bar{\Phi}_{\text{induced}}]$. This cache multiplier multiplies the queue factor $(1-\rho_P)/(1-\rho'_P)$ of Theorem 3.8; it does not replace it. A multiplier common to all sessions scales every u_i^P by the same factor: it leaves the eviction order unchanged and moves only the comparisons with θ_P (keep or drop, admit).

6.8 What is exact and what is approximate

The list of Section 5.5 applies. In addition, Propositions 6.1 to 6.8 are exact given the model. The new modelling choices are the link as fair bandwidth sharing (a transfer engine that sends in arrival order would add a head-of-line term to Proposition 6.1) and a single survival law with the mean wait in the

feedback map. The link constants x_0 , κ and B must be measured, and the survival law G estimated from the workload.

6.9 Summary

- A transfer miss is priced exactly by insensitivity and grows linearly in the context; a prefill miss grows quadratically and has a head-of-line term.
- Each pool has its own shadow price, and a decode cache is worth only the transfers it saves.
- The hit rate is also an output: overload can add an all-miss equilibrium. An admission cap removes the overload cause, but the pool channel can still leave one.

6.10 Exercises

Exercise 6.1. Derive (6.3)–(6.5) from Section 6.1. Why can turn k be admitted only after turn $k - 1$, and why does it reach the prefill server in arrival order? Which steps take zero time? *Hint:* admission is in arrival order, and a FIFO stage serves only its first turn.

Exercise 6.2. Disable prefix caching in the model of Section 6.1. Show from Section 6.1 that every turn then has $H_k = 0$, and compute $\mathbb{E}[S]$ from (6.6). Which of the conditions (6.8) becomes harder to meet?

Exercise 6.3. A session with context K is paused at both instances. Using Proposition 6.2, show that for K large enough it is kept at the prefill instance whenever it is kept at the decode instance, if $\theta_P = \theta_D$ and $b > 0$. *Hint:* $\Phi_i \geq \Delta S_i \geq bK^2/2$ grows quadratically, Φ_i^X linearly in K .

Exercise 6.4. Prove Proposition 6.8: h_δ solves $h = (1 - \delta)(\nu + kh)$. Then take $k = 0.5$ and $h_0 = 0.8$. Force $\delta = 0.1$ of the turns to miss. Compute h_δ and compare the loss $h_0 - h_\delta$ with the direct loss δh_0 . How does the ratio behave as $k \uparrow 1$? *Hint:* the ratio is $1/(1 - (1 - \delta)k)$.

Chapter 7

Metastability of a Prefix Cache Under Load

Section 6.7 found the hit rates a replica can sustain as fixed points, and Proposition 6.6 showed that there can be two. This lecture asks the dynamic question: does a burst of turns push a replica into a degraded state that outlasts the burst? Alvaro et al. [1] call such a state a **metastable failure** and study it for servers with timeouts and retries, through a Markov chain, its drift field, its slow eigenvalues and a calibration against a simulator. Here the loop is

longer queue $\rightarrow$ prefixes evicted while waiting $\rightarrow$ recompute $\rightarrow$ slower service $\rightarrow$ longer queue.

The answers depend on three engine rules: the order in which waiting turns are admitted, the eviction rule of the prefix cache, and the moment a turn’s working memory is taken. Section 7.2 proves what LRU keeps under first-come-first-served admission, Section 7.3 what that implies for a closed replica and its admission cap, Section 7.4 why the memory should be taken at admission, Section 7.5 what state a predictive model must keep, and Section 7.6 checks the predictions against vLLM’s scheduler rules in serQ. Section 7.7 collects the conclusions.

7.1 The replica

Replica with a prefix cache. Sessions submit turns to one replica. A session has at most one turn outstanding. Waiting turns are admitted first come first served (FCFS), at most B at a time. The prefix cache holds the prefixes of at most C sessions, all of the same size, and a session’s prefix enters it only when the session’s turn is served. A turn whose prefix is resident is a **hit** and costs S^{hit} of prefill; otherwise it is a **miss** and costs $S^{\text{miss}} \geq S^{\text{hit}}$. An admitted turn also holds a units of working memory, in units of one prefix.

vLLM’s scheduler approximates this model. It admits waiting requests in arrival order up to `max_num_seqs`, keeps a finished request’s blocks in an LRU free queue, and evicts from that queue when an admission needs blocks; the blocks of a waiting request stay evictable until it is admitted (serQ’s `docs/language.md`, §7, maps each rule to `scheduler.py` and `kv_cache_manager.py`). Two differences matter below. The cache shares its pool with the running requests’ growing KV, so the effective C is at most the pool; we take the whole pool, which is close when the admitted requests add little memory. And the hit is decided when a request is admitted, while its blocks become the most recent when it finishes, so for the recency order “served” means finished. Caching is by block, not by prefix.

Remark 7.1 (The correspondence with retries)

In [1] a state is a pair (u, v) : u requests in the server and its queue, and v requests in an *orbit* waiting to be retried. A timeout, more likely when u is large, turns a request into extra *arrivals*. Here an eviction, more likely when the queue is long, turns a hit into extra *work*, and the number of turns is conserved. The role of the orbit is taken by the waiting turns whose prefix is gone (Section 7.5).

7.2 What LRU keeps under FCFS

Number the services $t = 0, 1, 2, \dots$ and let x_t be the session served at t .

Under LRU a session finds its prefix exactly when fewer than C distinct sessions have been served since its last turn: LRU's cache is the head of the recency list, and every other session served moves in front (Mattson's stack-distance criterion [17]). Under FCFS the turns ahead of a waiting turn belong to distinct sessions, since a session has at most one turn outstanding, and all of them are served first. So a turn that joins behind C waiting turns misses, whatever the think times or the arrival process. When the queue is saturated, every session waits behind all the others and is served once per round.

Proposition 7.2 (Saturated rounds)

Let $N > C \geq 1$ sessions be served round-robin.

- (i) Under LRU, from an empty cache, every turn is a miss.
- (ii) Under any eviction rule, at most C of any N consecutive turns are hits.
- (iii) Pinning the prefixes of C sessions and caching no other attains the bound: C hits in every N consecutive turns.

Proof. (i) Between two services of a session, the other $N - 1 \geq C$ sessions are served, so it is not among the C most recent. (ii) Only a served session's prefix enters the cache, and the N turns belong to distinct sessions, so a turn that hits must find a prefix that was resident when the N turns began; there are at most C . (iii) The pinned sessions hit on each of their turns, and each is served once in every N consecutive turns. $\square$

LRU is therefore the worst rule under saturation, and it falls to zero even when the cache misses only one session ($C = N - 1$). The session just served is the one needed last, and LRU keeps it. A rule that evicts by the price of a miss per byte-second, $u_i = p_i \Phi_i / (c_i \tau_i)$ of the paper's scheduler, gives the just-served session the longest time to its next use τ_i and so the lowest price. Protecting a fixed set of sessions, rather than the most recent ones, is optimal here.

7.3 The closed replica: the congested queue and an admission rule

Project the replica onto the number n of waiting turns, with the mean service time $s(n) = S^{\text{hit}} + m(n)(S^{\text{miss}} - S^{\text{hit}})$ of a turn served at queue length n . The queue then moves by one turn at a time, and its law is that of a birth-death chain (Theorem 2.6), read in serving terms as follows.

birth-death chain	replica
level n	turns outstanding: waiting or in prefill
birth at rate λ_n	a turn arrives: a session ends its think time or tool call; $\lambda_n = \lambda$ (open), $(N - n)/Z$ (closed)
death at rate $\mu_n = 1/s(n)$	the engine finishes a turn's prefill; slower when more turns miss
$\pi(n)$	fraction of time with n turns outstanding
mode, trough	queue length the replica settles at (empty, congested); the tipping point between
$T_k, R(m, n)$	mean time for the queue to shrink by one from k ; to drain from m to n after a burst
barrier ΔV	how persistent the congested queue is

Lemma 7.3 (Birth-death toolkit)

For arrival rates λ_n and completion rates μ_n on $\{0, \dots, M\}$, positive where they are used:

- (i) $\pi(n+1) \geq \pi(n)$ if and only if $\lambda_n \geq \mu_{n+1}$: the queue lengths the replica settles at are those where arrivals stop outpacing completions, and the tipping points those where they start to;
- (ii) the mean time T_k for the queue to shrink from k to $k-1$ is $\sum_{j \geq k} \pi(j)/(\pi(k)\mu_k)$, and the drain time $R(m, n) = \sum_{k=n+1}^m T_k$ is the mean time to reach n waiting turns from m ;
- (iii) $R(m, n)$ depends only on the rates at queue lengths above n , and does not grow when completions there get faster or arrivals there slower;
- (iv) $T_k \geq \pi(j)/(\pi(k)\mu_k)$ for every $j \geq k$: draining past a tipping point k takes at least $e^{\Delta V}/\mu_k$, where $\Delta V = \log(\pi(b)/\pi(k))$ is the barrier to a congested queue length b above it (the lower half of the Arrhenius law).

Proof. (i) $\pi(n+1)/\pi(n) = \lambda_n/\mu_{n+1}$ by Theorem 2.6. (ii) The first-step equations $\mu_k T_k = 1 + \lambda_k T_{k+1}$, $\mu_M T_M = 1$ have a unique solution by backward induction, and the formula satisfies them by the cut equation $\pi(k)\lambda_k = \pi(k+1)\mu_{k+1}$; the chain is skip-free, so passage times add. (iii) The ratios $\pi(j)/\pi(k)$, $j \geq k$, involve only rates at levels $\geq k$, and each factor λ_i/μ_{i+1} is monotone. (iv) Keep one term of the sum in (ii). $\square$

A replica serves a closed population: N admitted sessions, each thinking for a mean time Z between its turns. Under FCFS and LRU a turn that joins behind C waiting turns misses (Section 7.2). In the queue-length model this becomes the assumption that the engine completes turns at the all-miss rate $1/S^{\text{miss}}$ above queue length C .

Theorem 7.4 (The congested queue of an FCFS-LRU replica)

Let $\lambda_n = (N - n)/Z$ and $\mu_n = 1/S^{\text{miss}}$ for $n > C$.

- (i) For $C \leq n < N$, $\pi(n+1) \geq \pi(n)$ if and only if $n \leq N - Z/S^{\text{miss}}$. So if $N - Z/S^{\text{miss}} \geq C$, the law on $\{C, \dots, N\}$ is largest at the congested queue length $\lfloor N - Z/S^{\text{miss}} \rfloor + 1$.
- (ii) If $N \leq C + Z/S^{\text{miss}}$, the law falls from n to $n+1$ at every $n > C$: there is no congested mode above $C+1$.

Proof. (i) By Lemma 7.3(i), $\pi(n+1) \geq \pi(n)$ if and only if $(N - n)/Z \geq 1/S^{\text{miss}}$. (ii) For $n > C$, $n > N - Z/S^{\text{miss}}$, so (i) gives a strict fall. $\square$

The congested length is what the saturated throughput leaves. When every turn misses, the engine completes $1/S^{\text{miss}}$ turns per second, so by Little's law only Z/S^{miss} sessions are thinking and the other $N - Z/S^{\text{miss}}$ wait. Part (ii) is an admission rule: admit at most $C + Z/S^{\text{miss}}$ sessions, the cache plus the sessions that the think time hides at the all-miss rate. The cache appears only through C ; what moves the bound otherwise is a faster recompute or a cheaper restore, which lowers S^{miss} . For serQ's replica of Section 7.6, with the pool counted as $C \approx 32.6$ contexts and $N = 40$, the rule binds at

$Z^* = (N - C)S^{\text{miss}} \approx 2.5$ s. The simulated hit rate is 0.65 at $Z = 3$ s, 0.31 at 2.5 s and 0.04 at 2 s, so Z^* falls inside the cliff; at Z^* serQ is already half way down, and the degradation starts above it. The rule marks the collapse, not its onset. It is also sensitive to C : the calibrated $C = 38$ of Section 7.6 would put Z^* below one second. The engine is batched and this model has one completion rate, so the agreement is a check of the order of magnitude, not a fit.

Remark 7.5 (Many sessions: the open limit)

When sessions are many and short, the replica sees turns arriving at a rate λ that does not depend on its queue. With $\mu_n = 1/S^{\text{miss}}$ for $n > C$ the queue then has a stationary law if and only if $\lambda S^{\text{miss}} < 1$, for every C : the tail of the product form is geometric with ratio λS^{miss} . Above that limit, for $1/S^{\text{miss}} < \lambda < 1/S^{\text{hit}}$, the uncongested state is metastable, and the cache changes only its lifetime, which grows exponentially in C (Exercise 7.1).

7.4 Admit, then hold

In a closed population of N sessions the chain is finite and always stable; the failure is a long stay in a congested mode. Where the working memory of a turn is taken decides whether that mode can exist. Reserved on arrival, the n turns present leave $C - an$ for cached prefixes. Taken at admission, they leave $C - a \min(n, B)$.

Proposition 7.6 (Admit, then hold)

Let the loss probability be $m(n) = g(\text{free cache})$ with g nonincreasing, and the closed arrival rate $\lambda_n = (N - n)/Z$.

- (i) Taking memory at admission never serves slower than reserving it on arrival, at any queue length. Hence every recovery time is at least as short, and the stationary weight of every queue length relative to the empty queue is at least as small.
- (ii) With memory taken at admission, the stationary law of the queue length has no trough above the batch cap B : above B it is unimodal, and any barrier lies at or below B .

Proof. (i) $C - a \min(n, B) \geq C - an$, so g and hence s are no larger; apply Lemma 7.3(iii) and the product form. (ii) For $n \geq B$ the free cache, hence μ_{n+1} , is the same constant μ . A trough at $n + 1$ needs $\lambda_n < \mu$ and $\lambda_{n+1} > \mu$ by Lemma 7.3(i), but $\lambda_{n+1} < \lambda_n$. $\square$

Part (ii) does not exclude a congested mode. It says that the saddle in front of one must lie at or below B , so that losses must already start while at most B turns hold memory. In practice admission-time memory can be bistable only if aB is a sizeable part of the cache; with a large enough a the law can have a congested mode above B behind a trough below it. Reserving on arrival needs no such condition, since the memory keeps growing with the queue, and there the congested mode is real and its recovery time grows exponentially with the population. With m logistic in n , centred at $N/4$, and $Z = 1.5Ns$, the barrier of the congested mode grows by about 0.21 per session (Table 7.1). At $N = 40$ two modes coexist for $22.0 \leq Z \leq 121.1$ s: the turn rate that triggers the collapse and the rate needed to recover from it differ by a factor of 5.5. The inverse spectral gap is shorter than the recovery, because it follows the faster of the two transitions, here the collapse.

7.5 What a predictive model must keep

The queue length is not the whole state. A model of the replica that keeps the cache is the following chain.

N	saddle	bad mode	barrier ΔV	recovery (s)	1/gap (s)
20	4	14	5.1	8500	129
40	8	28	9.3	1.0×10^6	3260
60	12	42	13.4	9.3×10^7	7.6×10^4
80	16	56	17.6	7.8×10^9	1.7×10^6
120	25	84	25.9	4.7×10^{13}	8.3×10^8
160	33	112	34.2	2.5×10^{17}	3.8×10^{11}

Table 7.1: Closed queue, $Z = 1.5N$, logistic m centred at $N/4$: barrier from the congested mode to the saddle, mean recovery time to the empty queue (Lemma 7.3(ii)), and the inverse spectral gap.

Queue-and-cache chain. There are N sessions. A state (h, m, c) counts the waiting turns whose prefix is resident (h) and lost (m), and the thinking sessions whose prefix is lost (c); $w = N - h - m - c$ think with a resident prefix, and $q = h + m$. Warm sessions return at rate w/Z as hits, cold ones at rate c/Z as misses. The queue is served by processor sharing, so hits complete at rate $h/(qS^{\text{hit}})$ and misses at $m/(qS^{\text{miss}})$, and a completed turn leaves a warm thinker. The idle prefixes $w + h$ must fit in $C - aq$: every waiting turn holds memory a , the case Proposition 7.6 warns against. While they do not fit, a waiting hit is evicted with probability $1 - \phi$ and a warm thinker with probability ϕ ($\phi = 1$ protects waiting prefixes).

With $N = 40$, $Z = 27$ s, $C = 48$, $a = 1$, $\phi = 0$, $S^{\text{hit}} = 0.5$ s and $S^{\text{miss}} = 5$ s, the chain has 12341 states and a closed class of 297: an overflow means $h > C - N$, so eviction caps the waiting hits at 8 and never evicts a thinker. The coordinate that matters is the number m of waiting misses. The queue-length law is bimodal, with probability 0.50 of at least 16 waiting turns.

Lemma 7.7 (Lumping preserves the stationary marginal)

Let X be an irreducible CTMC on a finite state space with stationary law π , and let $q(X)$, the number of waiting turns, change by at most one per transition. The birth–death chain whose arrival and completion rates at n are those of the full chain averaged over π given $q = n$ has the unique stationary law $p(n) = \pi(q = n)$.

Proof. Irreducibility gives $p > 0$ on the range of q and an irreducible lumped chain. Summing global balance over $\{q \leq n\}$ leaves only the flows between n and $n + 1$: $p(n)\bar{\lambda}_n = p(n + 1)\bar{\mu}_{n+1}$, which is detailed balance for the lumped chain (Lemma 2.3). $\square$

The lumped chain on the closed class reproduces the queue-length law to $3e - 16$, but not the dynamics. Its gap is 0.00726/s against the full chain’s slowest rate 0.00227/s, and it predicts a recovery from the congested region to $q \leq 1$ of 418 s against 1375 s. From the same empty queue the congested region has probability 0.005 after 30 s with a warm cache and 0.96 after a cache flush (Table 7.2). A burst of 30 s (think time 12 s) raises the probability of the congested region 600 s later from 0.35 to 0.60, and the replica needs 1415 to 1496 s to drain (Table 7.3). The waiting-miss count is the hidden slow variable, as the orbit is in [1]; a queue-length model is right for steady state and optimistic about recovery.

t (s)	$\mathbb{P}(q_t \geq 16)$	warm	cold	$\mathbb{E}q_t$	warm	cold
30		0.005	0.962		2.5	22.3
100		0.055	0.996		4.1	26.7
300		0.210	0.786		7.5	19.7
1000		0.439	0.554		12.4	14.9

Table 7.2: Queue-and-cache chain: same empty queue, warm cache ($c = 0$) or flushed cache ($c = N$). Exact transient laws.

burst (s)	$\mathbb{E}q$ at its end	$\mathbb{P}(q \geq 16)$ 600 s later	mean time to $q \leq 1$ (s)
0	0.0	0.35	0
10	11.8	0.47	727
30	26.2	0.60	1415
60	32.1	0.62	1488
120	33.2	0.62	1496

Table 7.3: Queue-and-cache chain, think time 27 s and 12 s during the burst.

The ranking of interventions also depends on the criterion (Table 7.4). A faster prefill gives the best response time, a slightly larger cache the fastest recovery. Speeding up only the miss path, which sets the stability limit of the open queue (Theorem 7.4), helps least in this closed population, where the onset of losses is fixed by $C - N$ and a and the hit work sets how fast the queue drains below it. Protection helps ($\phi = 1$ beats $\phi = 0.5$ beats $\phi = 0$) because the chain charges nothing for pinned memory.

arm	response (s)	rank	$\mathbb{P}(q \geq 16)$	recovery (s)	rank
base	14.02	7	0.498	1375	7
prefill $\times 1.25$ (hit and miss)	1.07	1	0.003	155	2
hit path $\times 1.25$	1.20	2	0.008	249	3
miss path $\times 1.25$ (recompute or reload)	6.63	6	0.223	530	6
KV capacity $C + 4$	1.55	3	0.002	114	1
protect waiting, $\phi = 1$	2.45	4	0.016	268	4
protect waiting, $\phi = 0.5$	4.04	5	0.089	439	5

Table 7.4: Queue-and-cache chain: the base chain and six single changes. Exact stationary laws and mean hitting times.

7.6 Experiments

We follow the main experiments of [1]: a failure in time, the drift field, recovery time against the initial state, the parameters where metastability appears, and a calibration against a simulator. The simulator is serQ running a closed-loop program on vLLM’s scheduler rules (FCFS admission of at most 16 requests, an LRU block cache, continuous batching): 40 sessions with a 16 000-token context each and a pool of about 33 contexts. All numbers are model or simulation output, not measurements.

A metastable failure. Figure 7.1 follows a replica through normal load, a 60 s burst and normal load again (think time 42.5 s, 12 s during the burst) in the queue-and-cache chain. When every waiting turn holds 1.2 contexts of memory, the burst fills the queue and the replica stays congested after the load has returned. The exact probability of at least 16 waiting turns is 0.93 at $t = 1000$ s and 0.60 at 3000 s, against 0.10 at 3000 s without the burst (the stationary value is 0.18). The hit share of completed turns collapses and recovers over thousands of seconds. When the same replica takes the memory at admission, with at most 16 turns admitted, the queue drains within several hundred seconds: the probability is 0.04 at 1000 s and 0.00 at 3000 s (stationary 0.00). This is the shape of the failures in [1], a trigger followed by a degraded state that outlives it. Here aB exceeds the slack $C - N$, so Proposition 7.6(ii) alone does not exclude a congested mode for the second replica; the exact laws show how small it is at this load.

Drift and recovery. The mass sits at the empty queue and in a band of 16 to 32 waiting misses with at most 8 waiting hits (Figure 7.2, left). From ten waiting turns the recovery takes 508 s with a

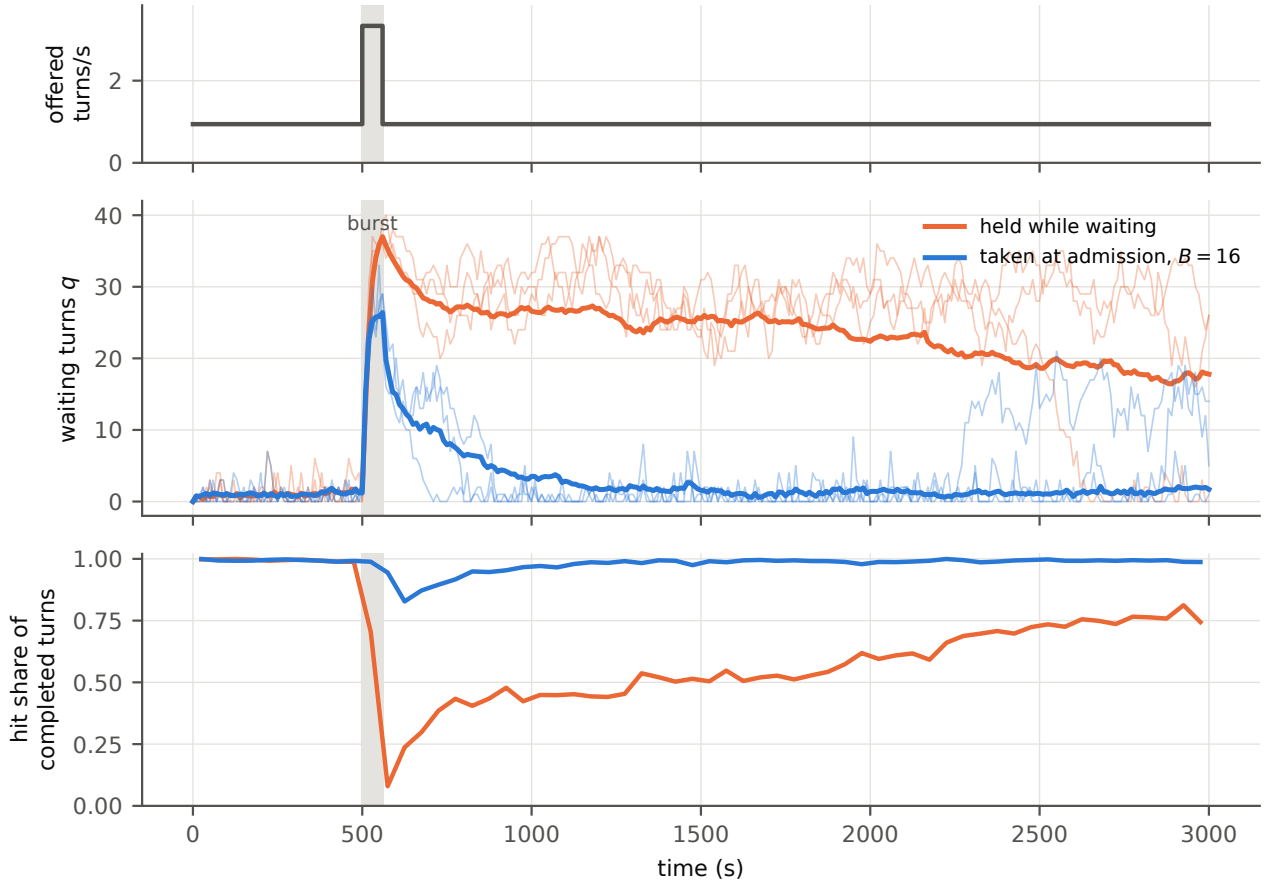


Figure 7.1: A metastable failure. Offered turn rate (top), waiting turns (middle: mean of 40 sample paths drawn by the Gillespie algorithm, three paths faint) and hit share of completed turns in 50s windows (bottom), for a queue-and-cache chain whose waiting turns hold memory and for the same chain with memory taken at admission. $N = 40$, $C = 48$, memory 1.2 contexts per turn. The grey band is the burst.

warm cache and 1552s after a flush, while the lumped chain predicts 202s; the time rises steeply over the first ten to fifteen turns and then saturates (right).

Where metastability appears. Figure 7.3 maps the slowest relaxation time over the memory a held per waiting turn and the think time. The queue-length law is bimodal only for $a \geq 1$, memory per waiting turn comparable to a whole context. An engine that allocates at admission has a small a for at most B turns. A disaggregated decoder that admits a request and allocates blocks for its whole prompt while it waits for the KV transfer (vLLM’s `WAITING_FOR_REMOTE_KVS`, `scheduler.py:1264–1294`, as cited in serQ’s `docs/language.md`) holds about a context for every request whose transfer is pending. Whether that makes it metastable is the prediction to test.

vLLM’s rules in serQ. serQ’s vLLM program allocates at admission with $a \approx 0.04$, far below what Figure 7.3 needs, so the prediction is a cliff without metastability (Figure 7.4). Here a hit means that the whole context was reused. From five seeds and a cold or a warm start (both from an empty queue), the hit rate falls from 0.77 at $Z = 4$ s to 0.00 at $Z = 1.5$ s and the mean TTFT rises from 0.31 to 7.0s; the two starts end within 0.020 of each other at every load. The cliff is consistent with Proposition 7.2(i): near saturation the turns are admitted nearly round-robin and LRU evicts the next one in line. The bursts are the congested starts, and they show no hysteresis either. After a burst at $Z = 1.5$ s the hit rate, 0.00 during the burst, is back within 0.05 of its baseline 65s after a 60s burst

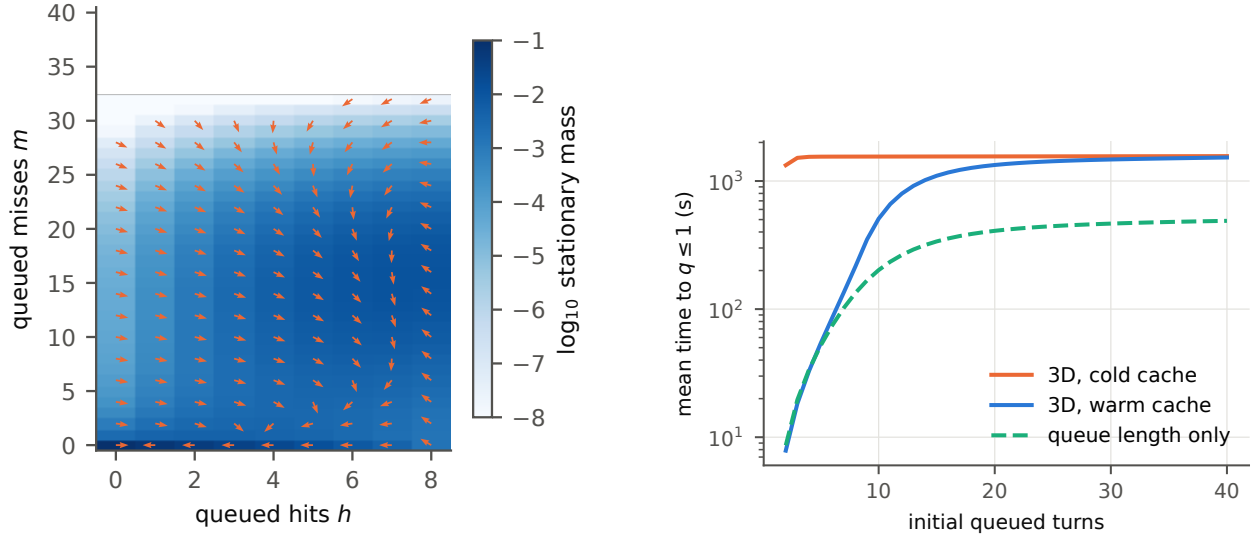


Figure 7.2: Queue-and-cache chain ($a = 1$). Left: mean drift of the waiting hits and misses (arrows, direction only) over the stationary mass. Right: mean time to $q \leq 1$ against the initial queue, from as many waiting hits as the cache holds (at most 8) with a warm cache, from waiting misses after a flush, and in the lumped chain.

and 50s after a 300s burst (Figure 7.5b), against more than a thousand seconds in the chain with $a = 1$ (panel a).

Calibration. We fit a chain to serQ’s hit-rate sweep and then compare the bursts, which the fit does not see. Service times come from serQ’s engine utilisation at two loads, 0.025s per hit and 0.336s per miss; memory is counted in contexts, $C \approx 32.6$, an admitted turn adds $a = 0.04$, and an admitted miss holds the prefix it recomputes. Each chain is fitted over a grid of two parameters. The chain of Section 7.5, fitted over C and a , cannot follow serQ (Figure 7.6a): its best error in hit rate is 0.36, and at the highest load it predicts 0.63 against serQ’s 0.00. Under processor sharing a short hit overtakes the misses, and every waiting turn is served at once although only 16 are admitted. A chain that serves one turn at a time, picks the next at random among the waiting turns unless an evicted turn waits at the head, and makes a waiting prefix LRU’s victim with weight β against a thinking one, fits better: $C = 38$, $\beta = 12$, error 0.12. The fitted C is above the physical 32.6, β is at the edge of its grid (largest value 12), the best processor-sharing fit has $C = 38$ and $a = 0.5$, and at the highest load the chain still predicts 0.16 against 0.00, so the fit is partial. On serQ’s 25s windows it recovers from the held-out bursts in 65s and 75s against serQ’s 65s and 50s (panel b): matching the shorter burst and one window slower for the longer burst. During the burst its hit share falls to 0.16 against serQ’s 0.02. A model of vLLM’s rules must couple LRU’s victim to the next turn served, which processor sharing cannot do.

7.7 Conclusions

1. *FCFS with LRU loses every prefix behind a queue of C .* A turn behind C others misses (Section 7.2); under saturation LRU hits nothing, any rule at most C of N per round, and pinning C sessions attains C (Proposition 7.2). The cliff in serQ’s run of vLLM’s rules is consistent with this.
2. *Admit at most $C + Z/S^{\text{miss}}$ sessions.* Behind the cache a closed FCFS–LRU replica congests at about $N - Z/S^{\text{miss}}$ outstanding turns, and with at most $C + Z/S^{\text{miss}}$ sessions it has no congested

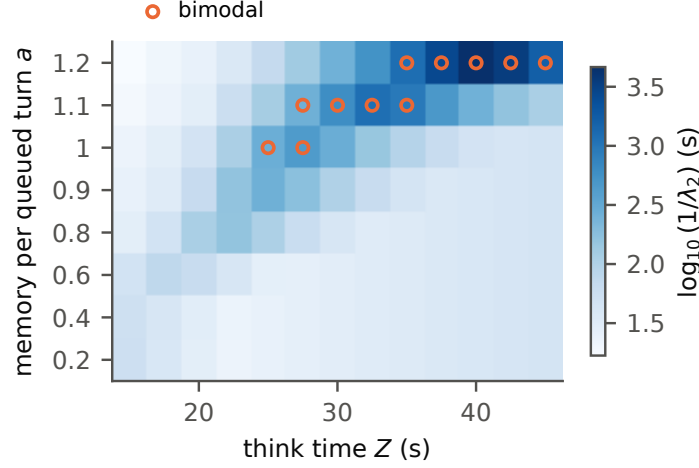


Figure 7.3: Queue-and-cache chain, $N = 40$, $C = 48$: $\log_{10}$ of the slowest relaxation time; circles mark a bimodal queue-length law.

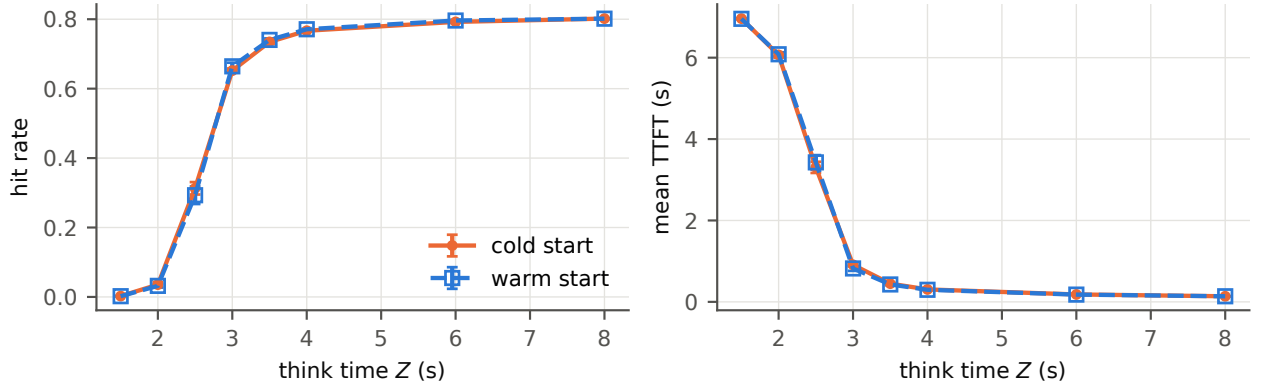


Figure 7.4: serQ, vLLM’s rules: hit rate and mean TTFT over 2000–4000 s against the think time, from a cold and a warm cache (five seeds, mean and standard deviation).

mode there (Theorem 7.4). For serQ’s replica the rule binds at $Z^* \approx 2.5$ s, inside the simulated cliff; the agreement depends on counting the whole pool as C . Faster prefill or a cheaper restore raises the cap; in the open limit the same miss work sets the stability limit $\lambda S^{\text{miss}} < 1$.

3. *Take memory at admission.* Doing so never slows service or recovery, and it puts every barrier at or below the batch cap, so bistability needs the admitted turns alone to start the losses (Proposition 7.6). After the same 60 s burst, a replica whose waiting turns hold memory is still congested with probability 0.60 after 3000 s; the same replica with memory taken at admission has drained (Figure 7.1). Metastability appears when every waiting turn holds about a context of memory (Figure 7.3), and serQ’s admission-time engine shows a cliff without hysteresis (Figure 7.4). Engines that hold memory while a request waits, such as a decoder waiting for a KV transfer, are the ones to test.
4. *A predictive model must keep the waiting misses and couple LRU’s victim to the next turn served.* Queue length alone is exact for steady state and optimistic about recovery by a factor of about three (Section 7.5). A processor-sharing chain cannot be calibrated to vLLM’s rules; a chain that serves one turn at a time and evicts the next in line fits partly and recovers from held-out bursts in 65 and 75 s where serQ takes 65 and 50 s.
5. *Rank interventions by the question asked.* Steady state and recovery rank them differently (Ta-

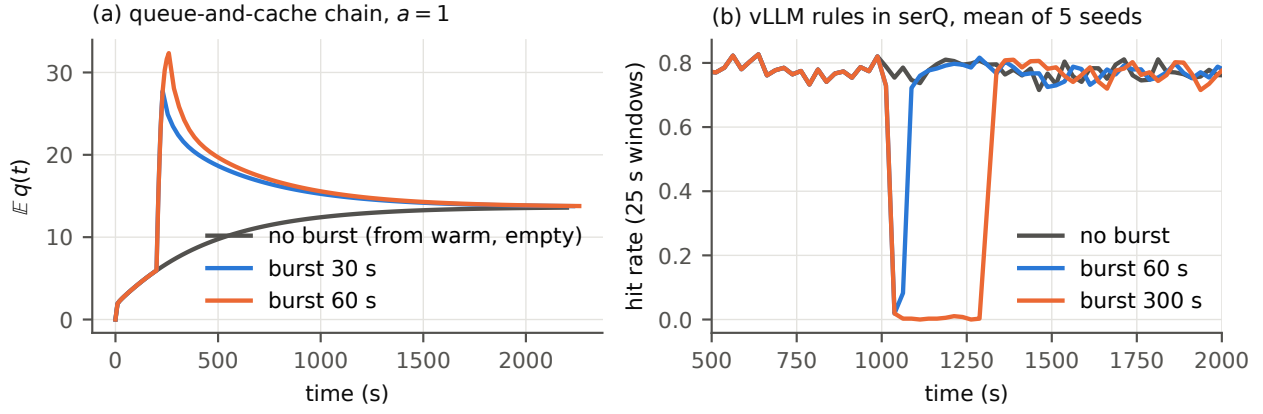


Figure 7.5: Bursts. (a) Queue-and-cache chain, $a = 1$, think time 27s and 12s during the burst, exact transient means. (b) serQ, think time 4s and 1.5s during the burst, hit rate in 25s windows, mean of five seeds.

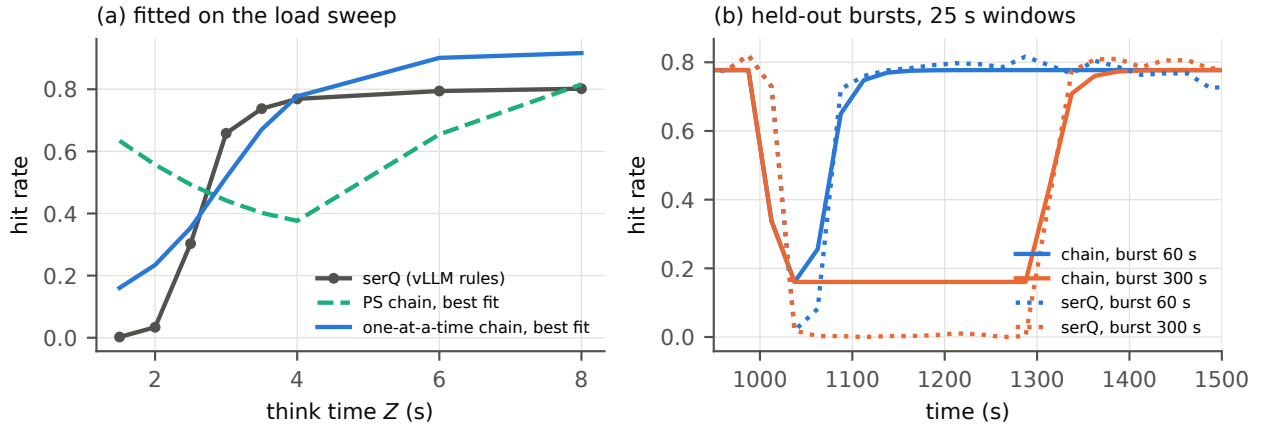


Figure 7.6: Calibration against serQ. (a) Full-reuse hit rate against the think time: serQ and the best fits of the processor-sharing and the one-at-a-time chains. (b) Held-out bursts on 25s windows.

ble 7.4): the best response time comes from a faster prefll, the fastest recovery from more cache.

7.8 What is exact and what is approximate

- *Proved* (and checked in Lean, module `CacheOrder` and `Metastability`, except where noted): Proposition 7.2, the LRU criterion of Section 7.2 (its FCFS step is in the prose), Lemma 7.3 (its algebraic content; that T_k and R are hitting times is proved here only), Theorem 7.4 and Proposition 7.6, and Lemma 7.7 (here only). Every chain number is computed from the generator; the sample paths of Figure 7.1 are drawn from it for the picture only.
- *Simulation*: the serQ runs model vLLM’s scheduler; they are not measurements of an engine.
- *Modelling choices*: prefixes of equal size and session-level LRU instead of blocks; the queue-length model of Theorem 7.4, in which the loss depends on the current queue rather than the queue a turn joined, and a single completion rate in place of a batched engine; exponential service; processor sharing or one turn at a time instead of a batched engine; no cost of pinned memory.
- *To be measured*: the loss onset, the slack $C - N$ and the memory per waiting turn on a real engine, with the burst protocol (normal load, a short burst, normal load; warm and flushed starts).

7.9 Exercises

Exercise 7.1. Let $m(n) = \mathbf{1}\{n \geq C\}$ in the open queue, with $1/S^{\text{miss}} < \lambda < 1/S^{\text{hit}}$. Show that the fluid saddle is at C , and that the mean time to reach C from the empty queue is of order $(\lambda S^{\text{hit}})^{-C}$.

Exercise 7.2. In Proposition 7.2(i) let each round serve the N sessions once, in an order that may change from round to round. Show that LRU can then hit, and find the largest number of hits in a round as a function of N and C .

Exercise 7.3. Add a concurrency cost to the queue-and-cache chain of Section 7.5: pinned prefixes slow every turn by the factor $1 + \gamma \phi h/N$. Find, numerically, a γ at which an interior ϕ minimises the recovery time.

Appendix A

Executable Examples in serQ

The examples accompanying these notes are programs in serQ v0.1.6 (IR version 14, commit 5d7d984). A serQ program is a deployment, which declares memory pools, service stages and a server block, run under a workload, a file of its own (in `programs/workloads/`) that declares the sessions and what each request sends. Running it simulates the deployment; it does not measure a real serving system. Each stage kind is one of the queues of the notes.

serQ	Queue	Where
<code>stage s : fifo;</code>	single FIFO server	Chapter 3
<code>stage s : ps(f);</code>	processor sharing with capacity f	Chapter 2
<code>stage s : delay;</code>	delay station	Chapter 4
<code>engine E {...}</code>	engine iterations with a token budget	Chapter 5
<code>hold, reserve</code>	a pool allocation and its admission gate	Section 2.3
<code>transfer</code>	KV transfer over a link	Chapter 6

For example, `stage decode : ps(min(present, 16));` is a batch whose capacity grows with the number `present` of turns up to 16. An engine schedules tokens per iteration: its `scheduler` advances the running requests and admits waiting ones within a token `cap`, in an order it states, and its `worker` gives the iteration's time (`forward`); a FIFO prefill order must be chosen there, since chunking alone does not give it.

The programs follow engine rules that the analytical models leave out. The executable `transfer` reserves the destination before it releases the source, unlike Section 6.1. Allocations round up to whole blocks and cached prefixes round down, so a partial hit has its own work law (Remark 5.2). A preempted decode keeps its generated tokens and re-prefills only lost KV. Ending a session does not remove its cached blocks.

A.1 What to run and what it checks

`programs/lecture_pd.sq` implements Section 6.1; it records the TTFT at the prefill instance (`ttft`, Eq. (6.7)), the TTFT a client sees after the hand-over (`ttft_client`: the transfer and the wait for decode memory lie between the two) and the transfer itself; `programs/vllm.sq` implements an engine with vLLM's rules. From the repository root, `make lecture-results` runs the course checks, which compare simulations with Pollaczek–Khinchine, PS insensitivity, mean value analysis, the disaggregated bottlenecks, the equilibria of Example 6.7, and the price bracket on a step engine. Passing checks support the formulas in their stated models; they do not make a program a proved queue.

A corrected result. An earlier version of these notes ran the disaggregated example on the preceding serQ release, where the decode capacity used a name that the workload also used for a token count. Every decode then received the full capped throughput even when alone, which is impossible:

mean decode work is $200 \times 0.0002 = 0.04$ s, yet the decode took about 0.0025 s. The historical table records simulated means over three matched seeds on the releases named in its header.

Metric (seconds)	Previous runtime	serQ v0.1.1
TTFT (prefill instance)	0.04327	0.04357
Turn response	0.05117	0.08868
Decode service	0.00249	0.03969

Bibliography

- [1] P. Alvaro, R. Isaacs, R. Majumdar, K.-K. Muniswamy-Reddy, M. Salamati, S. Soudjani, Formal analysis of metastable failures in software systems, arXiv:2510.03551v2, 2025.
- [2] S. Asmussen, *Applied Probability and Queues*, 2nd ed., Springer, 2003.
- [3] F. Baskett, K. M. Chandy, R. R. Muntz, F. G. Palacios, Open, closed, and mixed networks of queues with different classes of customers, *Journal of the ACM* 22(2), 1975.
- [4] J. Csirik, J. B. G. Frenk, M. Labbé, S. Zhang, Heuristics for the 0-1 min-knapsack problem, *Acta Cybernetica* 10, 1991.
- [5] G. B. Dantzig, Discrete-variable extremum problems, *Operations Research* 5(2), 1957.
- [6] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, Cambridge University Press, 2013.
- [7] A. R. Karlin, M. S. Manasse, L. Rudolph, D. D. Sleator, Competitive snoopy caching, *Algorithmica* 3, 1988.
- [8] F. P. Kelly, *Reversibility and Stochastic Networks*, Wiley, 1979.
- [9] A. Khinchine, Mathematical theory of a stationary queue, *Matematicheskii Sbornik*, 1932.
- [10] A. Y. Khinchine, *Mathematical Methods in the Theory of Queueing*, Griffin, London, 1960.
- [11] J. F. C. Kingman, *Poisson Processes*, Oxford University Press, 1993.
- [12] L. Kleinrock, *Queueing Systems, Volume 2: Computer Applications*, Wiley, 1976.
- [13] L. Kleinrock, *Queueing Systems, Volume 1: Theory*, Wiley, 1975.
- [14] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik, *Quantitative System Performance*, Prentice-Hall, 1984.
- [15] J. D. C. Little, A proof for the queueing formula $L = \lambda W$, *Operations Research* 9(3), 1961.
- [16] R. M. Loynes, The stability of a queue with non-independent inter-arrival and service times, *Proc. Cambridge Philosophical Society* 58(3), 1962.
- [17] R. L. Mattson, J. Gecsei, D. R. Slutz, I. L. Traiger, Evaluation techniques for storage hierarchies, *IBM Systems Journal* 9(2), 1970.
- [18] P. Naor, The regulation of queue size by levying tolls, *Econometrica* 37(1), 1969.
- [19] J. R. Norris, *Markov Chains*, Cambridge University Press, 1997.
- [20] F. Pollaczek, Über eine Aufgabe der Wahrscheinlichkeitstheorie, *Mathematische Zeitschrift*, 1930.
- [21] M. Reiser, S. S. Lavenberg, Mean-value analysis of closed multichain queueing networks, *Journal of the ACM* 27(2), 1980.
- [22] S. M. Ross, *Stochastic Processes*, 2nd ed., Wiley, 1996.

- [23] A. Agrawal et al., Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve, 2024.
- [24] S. Stidham, A last word on $L = \lambda W$, *Operations Research* 22(2), 1974.
- [25] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific Journal of Mathematics* 5(2), 1955.
- [26] R. W. Wolff, Poisson arrivals see time averages, *Operations Research* 30(2), 1982.