Split-Merge Model of Workunit Replication in Distributed Computing

Alexander Rumyantsev† Srinivas Chakravarthy‡

†IAMR KRC RAS, ‡Kettering University

30.08.2017

Outline

You Can Skip This Ad in Several Seconds

Who we are:

- Research group supervised by Dr. Evsey Morozov
- Researchers from two labs: Laboratory for Telecommunications Systems (Dr. Andrey Pechnikov), Laboratory for Mathematical Cybernetics (Dr. Vladimir Mazalov)

What we do:

- Stability and performance of queueing models
- Exact analysis (stochastic processes, matrix-analytic method)
- Approximate/asymptotic analysis (bounds of performance)
- Stochastic simulation, Regenerative confidence estimation, Splitting techniques ...

This work is a joint research of the ongoing collaboration with Dr. Srinivas Chakravarthy from Kettering University, USA.

We are open for collaboration, and happy to help in solving practical tasks!

Some Basic Notions in Queueing Theory

Queueing system:

- open/closed,
- finite(absent)/infinite buffer,
- arrival process/distribution $T_i \sim F_T$,
- service times distribution $S_i \sim F_S$,
- number of servers,
- service discipline (e.g. FCFS),
- additional constraints (e.g. inventory requirements, number of cores),
- etc.

Kendall's notation. E.g. $M/M/c$ = exponentially distributed interarrival/service times, c identical servers.

Key Performance Characteristics of Queueing System

Workload: remaining work in the system (in terms of service time). E.g. for $G/G/c$ system: a vector of workload $W_i = (W_{i,1}, \ldots, W_{i,c})$.

Delay: time from arrival to the beginning of service, e.g. $D_i = \min_{j=1,\ldots,c} W_{i,j}$.

Latency: time from arrival to departure: $D_i + S_i$ (a.k.a. sojourn time).

Queue: number of tasks in the system/in the queue.

Throughput: number of departures per unit time.

Blocking probability: probability to arrive at the full system.

Idle probability: probability to arrive at empty system.

Mostly we study all these by obtaining the stable states probabilities.

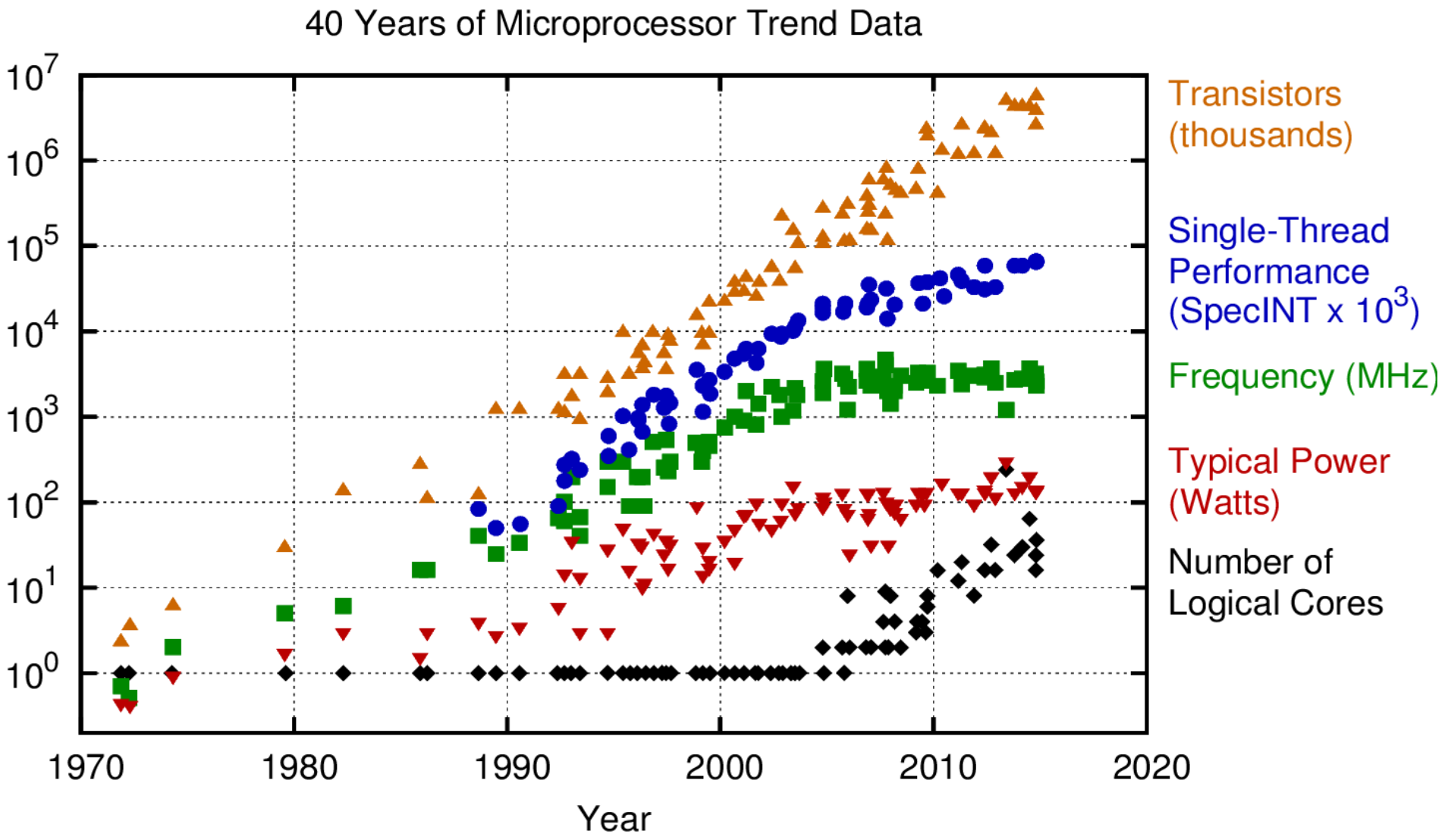
What is the Most Workload-Effective Queue?

Compare the total workload in $G/G/c$ queue (each server works at speed 1) with $G/G/1$ queue of one fast server (working at speed c), initially empty, fed with the same input of tasks.

$$cW_i^{(1)} \leqslant W_{i,1}^{(c)} + \cdots + W_{i,c}^{(c)}, \quad i \geqslant 1.$$

I.e. fast server queue performs like $G/G/c$ working at maximum capacity.
Moreover, single server latency is less, than multiserver if C.V. <1 [Brumelle, 1971]
Why don't we use the fast server systems?

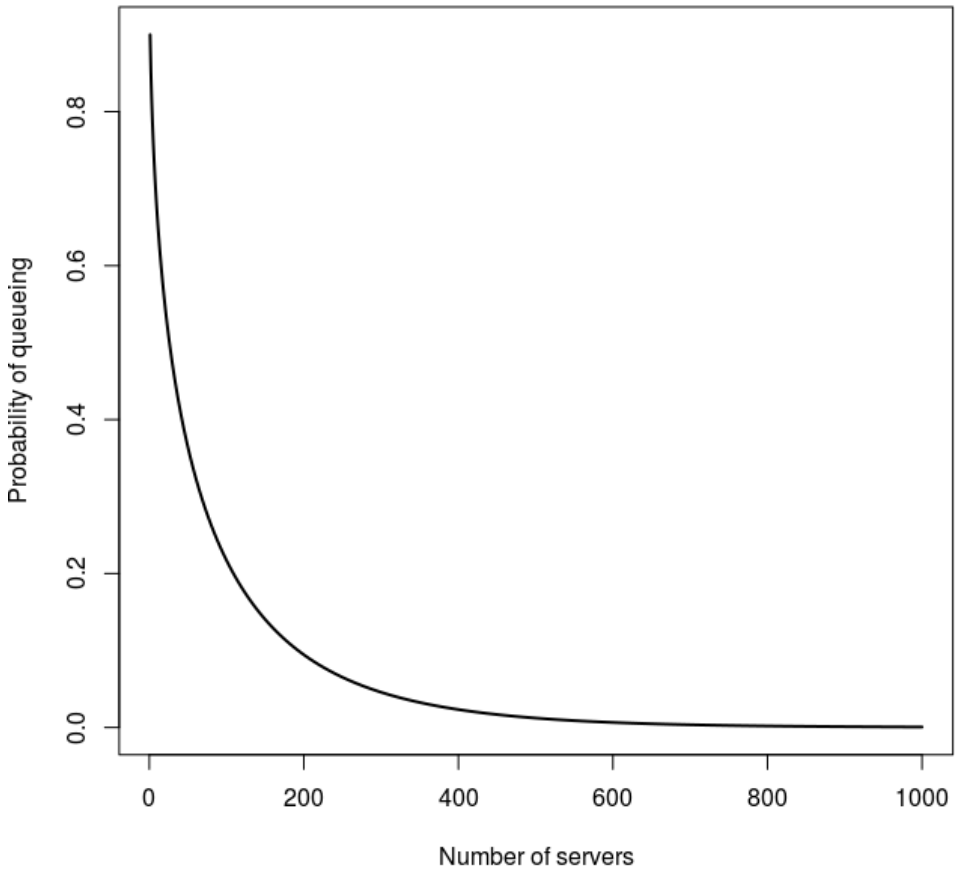
Free Lunch Is Over



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2015 by K. Rupp

Another Reason

Idle probability in multiserver system increases. E.g. consider $M/M/1$ system working at speed 1 vs. $M/M/c$ system with *slow* servers working at speed $1/c$.



Then probability of queueing is described by the Erlang C-formula.

$$C(c,\rho) = \left[1 + \left(1 - \frac{\rho}{c}\right)\frac{c!}{\rho^c}\sum_{k=0}^{c-1}\frac{\rho^k}{k!}\right]^{-1},$$

where $\rho = n\lambda/\mu$.
Moreover, the delay moment conditions become better for system with slow servers.

In some sense, the multiserver system is *more customer-friendly*.

What Do We Know About Multiserver System ...almost nothing

$G/G/c$ system, $c \geqslant 1$, arrival intensity λ, service intensity μ.

Stability condition

$$\rho := \lambda/\mu < c.$$

Does it imply emptying (number of clients in the system hitting zero)? Only if $c = 1$.
Workload of the system at arrival instants

Kiefer–Wolfowitz vector recursion

$$W_{i+1} = R(W_{i,1} + S_i - T_i, W_{i,2} - T_i, \ldots, W_{i,c} - T_i)^+.$$

Finite moments of workload components: for $j < \lceil \rho \rceil$ servers help each other

$$ES^{1+\frac{\alpha}{s-\lfloor \rho \rfloor}} < \infty \Rightarrow E\,[W_{\infty,j}]^\alpha < \infty.$$

For $j \geqslant \lceil \rho \rceil$ servers stuck with long tasks:

$$ES^{1+\frac{\alpha}{s-i}} < \infty \Rightarrow E\,[W_{\infty,j}]^\alpha < \infty.$$

Some Words on Stochastic Modeling of Multiserver System

Some points:

- Workload on arrivals: follow Kiefer–Wolfowitz.
- Delay of client i: $D_i = W_{i,1}$.
- Departure of client i: $d_i := t_i + D_i + S_i$.
- Workload on departures: introduce artificial arrivals at times d_i of zero-sized tasks.
- Queue size: a ± 1 at t_i, d_i.
- Or use the discrete event simulation.

Why I am talking about this? BOINC itself is a multiserver system. But with some distinctive features.

If You Decide to Do That (SM)

If you prefer R language: several ready packages:
`queueing`, `simmer`, `PDQ`, `arqas`, `queuecomputer`, `hpcwld`
(mostly for queueing systems, but also networks)

UseR!

ARENA and other general purpose simulators.

ns3 network simulator

BOINC specific: emBOINC (suspended since 2013)

Let Me Show You Something

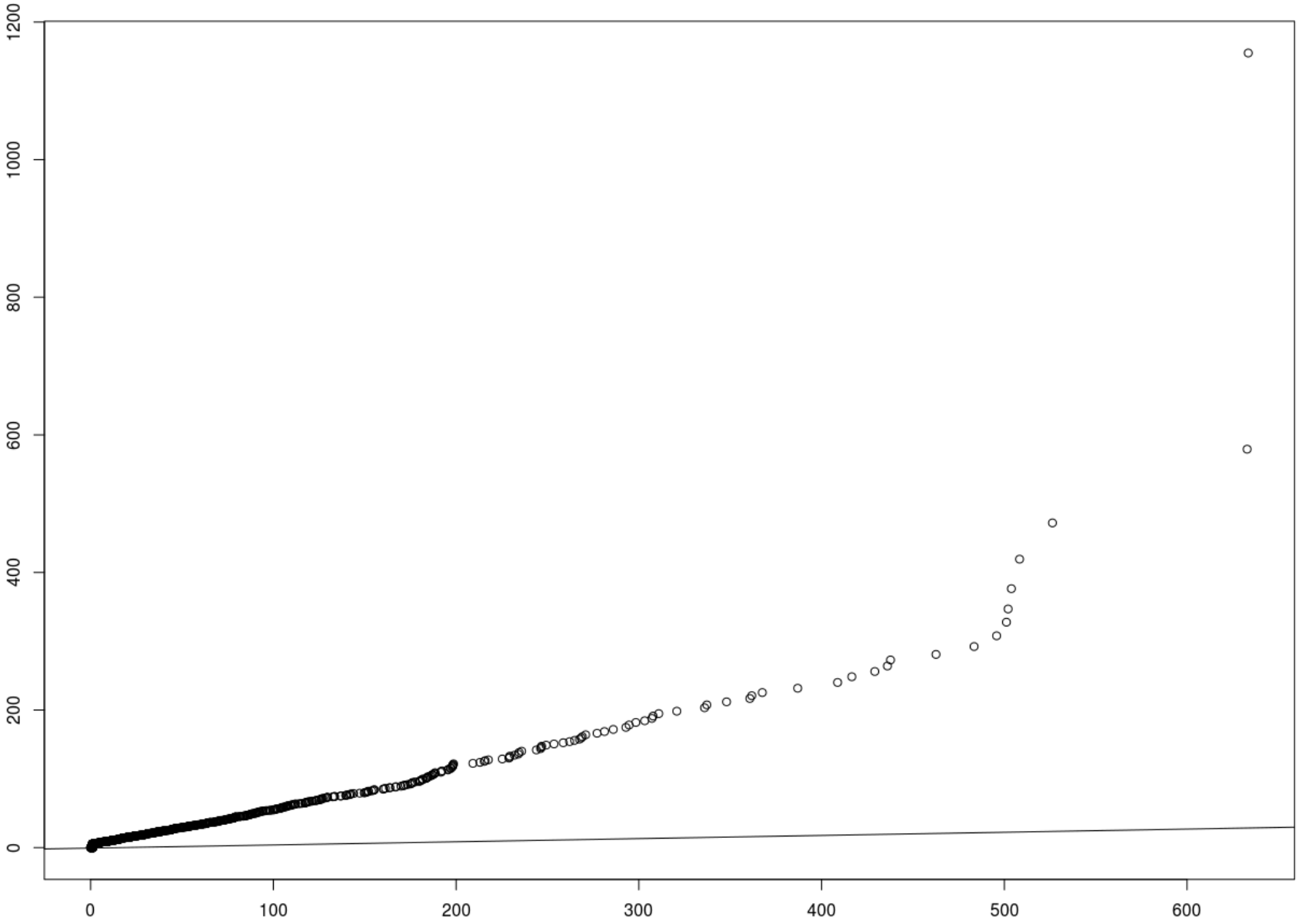
BOINC-related stuff: availability periods. Alternating 0-1 process (aka ON-OFF). Let ON-periods have F_{ON} (with mean μ_{ON}) and OFF-periods have F_{OFF} (with mean μ_{OFF}). Then sampling from stationary is as easy, as 1,2,3

1. Flip the coin: ON w.p. $\mu_{ON}/(\mu_{ON} + \mu_{OFF})$.
2. If ON, sample from $F_{ON}^{(e)}(x) = \frac{1}{\mu_{ON}} \int_0^x \overline{F}_{ON}(y)\, dy$.
3. If OFF, sample from $F_{OFF}^{(e)}(x) = \frac{1}{\mu_{OFF}} \int_0^x \overline{F}_{OFF}(y)\, dy$.

Thus, we start directly from *equilibrium*.
R implementation: package `ARPobservation` (3 sec on my desktop)

```
r_behavior_stream(n=1000,mu=20,lambda=1,F_event=F_weib(.5),
    F_interim=F_exp(),stream_length=100000)
```

Yes, They Do Differ



What So Special in $F^{(e)}$?

Equilibrium lifetime distribution (distribution of residual lifetime of renewal process), let's see its tail

$$\overline{F}^{(e)}(x) = \frac{1}{\mathrm{E}X} \int_x^\infty \overline{F}(y)\, dy = \frac{1}{\mathrm{E}X} \mathrm{E}(X - x)^+.$$

- For exponential distribution $\overline{F}(x) = \overline{F}^{(e)}(x)$.
- For heavy-tailed distribution $\overline{F}^{(e)}(x)/\overline{F}(x) \to \infty, \quad x \to \infty$.

In practice it means, that in a heavy-tailed case in most cases you face the *enormously long* period (e.g. workunit runtime).

Redundancy in Computing

Redundancy (aka Replication) is used in many fields of interest:

- RAID disk arrays: reduce failure probability
- Cloud, Map-Reduce: reduce latency
- BOINC: validation, speedup?

Extensively studied in many cases. Let's take a closer look.

Object

Introduce the (n, r, q) replication ($n \geqslant r \geqslant q$): send r replicas to n nodes, and wait for q to establish quorum.

1. We are able to cancel tasks.
2. Quorum eventually obtained.

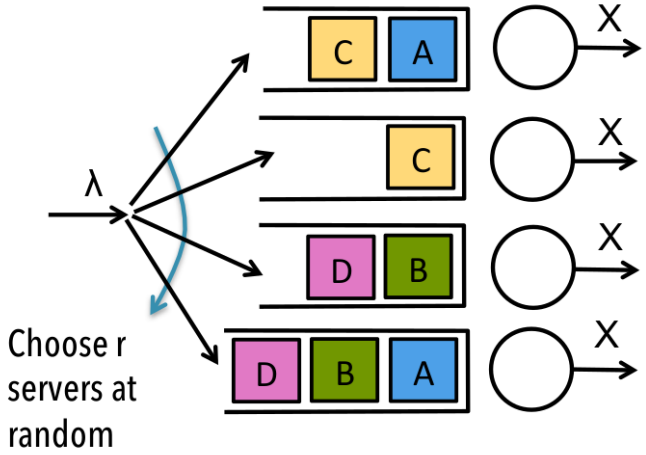
Fork-Join Model for Cloud

Recent series of works by Gauri Joshi.

Fork-Join

Tasks are dispatched (Forked) to servers, and are served independently. After service, they wait in the Join buffer.

Unnecessary replicas are killed either after obtaining quorum (Fork-Join), or at the beginning of service of q tasks (Fork-Early Cancel).



(image by G. Joshi)

Distribution of Order Statistics

Let $X_1, \ldots, X_n \sim F$, and $\overline{F} := 1 - F$, then
Minimum

$$P(X_{1:n} \leq x) = 1 - \overline{F}^n(x)$$

Maximum

$$P(X_{n:n} \leq x) = F^n(x)$$

In general,

$$P(X_{r:n} \leq x) = \sum_{i=r}^{n} \binom{n}{i} F^i(x) \overline{F}^{n-i}(x).$$

This easily generalizes to heterogeneous X_i.

What is Known for Fork-Join Models

We look at the tradeoff between the expected latency and expected cost (expected summary time) of computing.

(n,n,1) replication

System equivalent to $M/G/1$ queue with general service time $S_{1:n}$.
Latency: $\mathrm{E}S_{1:n} + \lambda \mathrm{E}S_{1:n}^2/(2 - 2\lambda \mathrm{E}S_{1:n})$. *Non increasing* for arbitrary F_S.
Cost: $n\mathrm{E}S_{1:n}$, non increasing (log-concave) /non decreasing (log-convex).

(n,r,1) replication

System equivalent to n/r independent $M/G/1$ queues with reduced input $\lambda r/n$ and general service time $S_{1:r}$.
Latency: $\mathrm{E}S_{1:r} + \lambda r \mathrm{E}S_{1:r}^2/(2n - 2\lambda r \mathrm{E}S_{1:r})$.
Cost: $r\mathrm{E}S_{1:r}$

(n,r,q) replication

Only some bounds known, in particular, by comparison with Split-Merge queue.

Surprisingly, in many practical cases either latency, or cost-optimal replication is

And Nothing Else Matters

The key is the log-concavity (log-convexity) of the tail, i.e. $\log \overline{F}$ is concave (convex).
For log-concave:

- $\mathrm{P}(X > x + t | X > t) \leqslant \overline{F}(x)$ (new-better-than-used)
- $\mathrm{E}(X - t | X > t)$ is non-increasing in t (mean residual life)
- $\mathrm{E}X \leqslant r\mathrm{E}X_{1:r}$ for integer $r \geqslant 1$, moreover $r\mathrm{E}X_{1,r}$ non-decreasing in r.

For log-convex: invert the signs.

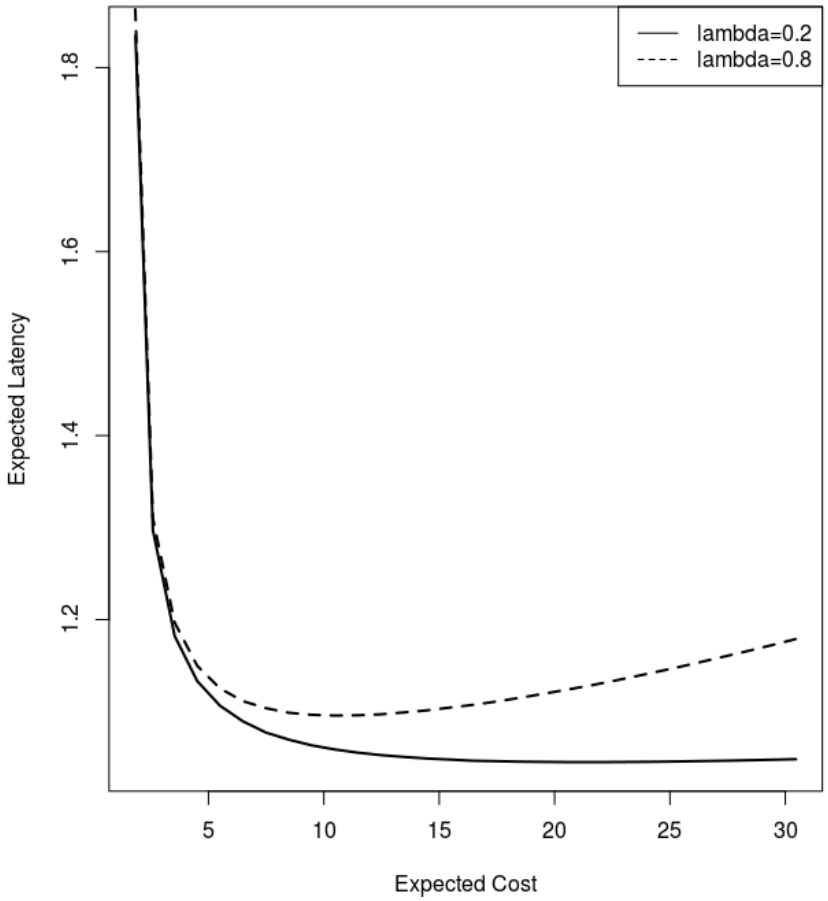
Log-concave: shifted exponential, uniform, Weibull with shape$\geqslant 1$, Gamma with shape$\geqslant 1$ etc.
Log-convex: Hyperexponential, Weibull with shape< 1, Gamma with shape< 1
Exponential: equality.
Pareto: neither, nor.

What's the difference

Log-concavity (convexity) characterizes the minimum, while heavy-tailedness characterizes the maximum of r.v.

Pareto case: rely on simulation

Recall
Latency:
$\mathrm{E}S_{1:r} + \lambda r \mathrm{E}S_{1:r}^2/(2n - 2\lambda r \mathrm{E}S_{1:r})$.
Cost: $r\mathrm{E}S_{1:r}$
If S is Pareto distributed, that is $\overline{F}_S(x) = x^{-\alpha}, x \geqslant 1, \alpha > 0$, then there is a tradeoff, which depends on λ.
(Simulation example with Pareto(2.5) - for group-based policy)



Why Split-Merge

Split-Merge has additional assumption: customer i doesn't start service before customer $i-1$ departs.

1. Upper bound for Fork-Join latency. Much easier to study.
2. Is applicable (with all it assumptions) to the EDG or RT BOINC concepts.
3. Some mathematical reasons below.

Mathematical Reasons

In fact, (n, r, q) Split-Match system is equivalent to $M/G/\lfloor n/r \rfloor$ queueing system with general service times $S_{q:r}$. But what class is closed under the ordering operations?
Phase-type distributions (PH). Benefits:

- Dense in the set of distributions (may approximate arbitrary distribution).
- Has some properties of exponential distribution (sojourn time in each phase is exponential).
- Developed methods of study/generation/fitting.
- May cover heterogenity.

Drawback: for the $S_{q:r}$ the underlying generator matrices become really huge. Is it useful for fast sample generation?
Then *rely on simulation* to get the order statistics, and evaluate the system as a multiserver queue (also covers heavy-tailed case etc.).

What Do I Mean By Really Huge

Denote $\mathcal{I}(n:k)$ the set of $n!/((n-k+1)!(k-1)!)$ lexicographically ordered $(n-k+1)$-tuples $(a_1,\ldots,a_{n-k+1})$ such that $a_1 < a_2 < \cdots < a_{n-k+1}$ and $a_i \in E_i$. Define

$$S_{k,n} = \operatorname{diag}\left(S_{j_1} \oplus \cdots \oplus S_{j_{n-k+1}}, (j_1,\ldots,j_{n-k+1}) \in \mathcal{I}(n:k)\right).$$

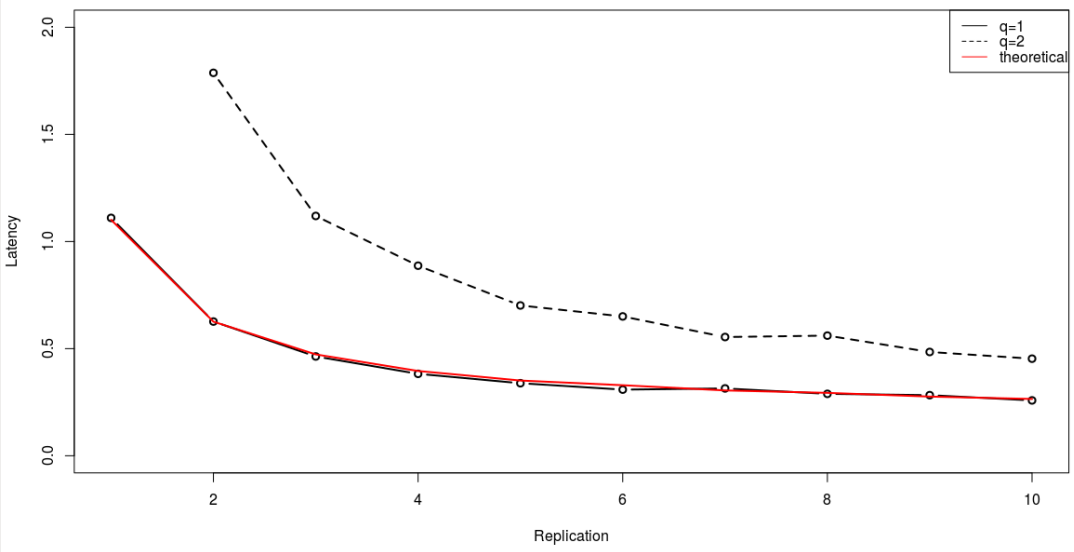
Define the matrix $S^0_{k,n} = -S_i\mathbf{1}$ of intensity of absorptions of only one of the $n-k+1$ chains. Then we obtain the iterative procedure to obtain $S_{k:n}$, for $k = 1,\ldots,n$:

$$T_{k:n} = \begin{pmatrix} S_{1,n} & S^0_{1,n} & 0 & 0 & \ldots & 0 \\ 0 & S_{2,n} & S^0_{2,n} & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & S_{k,n} \end{pmatrix}$$

Exponential (n, r, q) replication

Latency decreases (with r)... for free[1]!
All other relevant metrics do also (relevant to one fast server, recall, and note $r = 1$ is the classical multiserver system), even BOINC-specific, such as starvation.
The only thing, that gets worse, is the granted credit.
Only hope: $q = 1$ equivalent to $M/M/\lfloor n/r \rfloor$ model with service time $\sim \mathrm{Exp}(r\mu)$, thus, latency equals

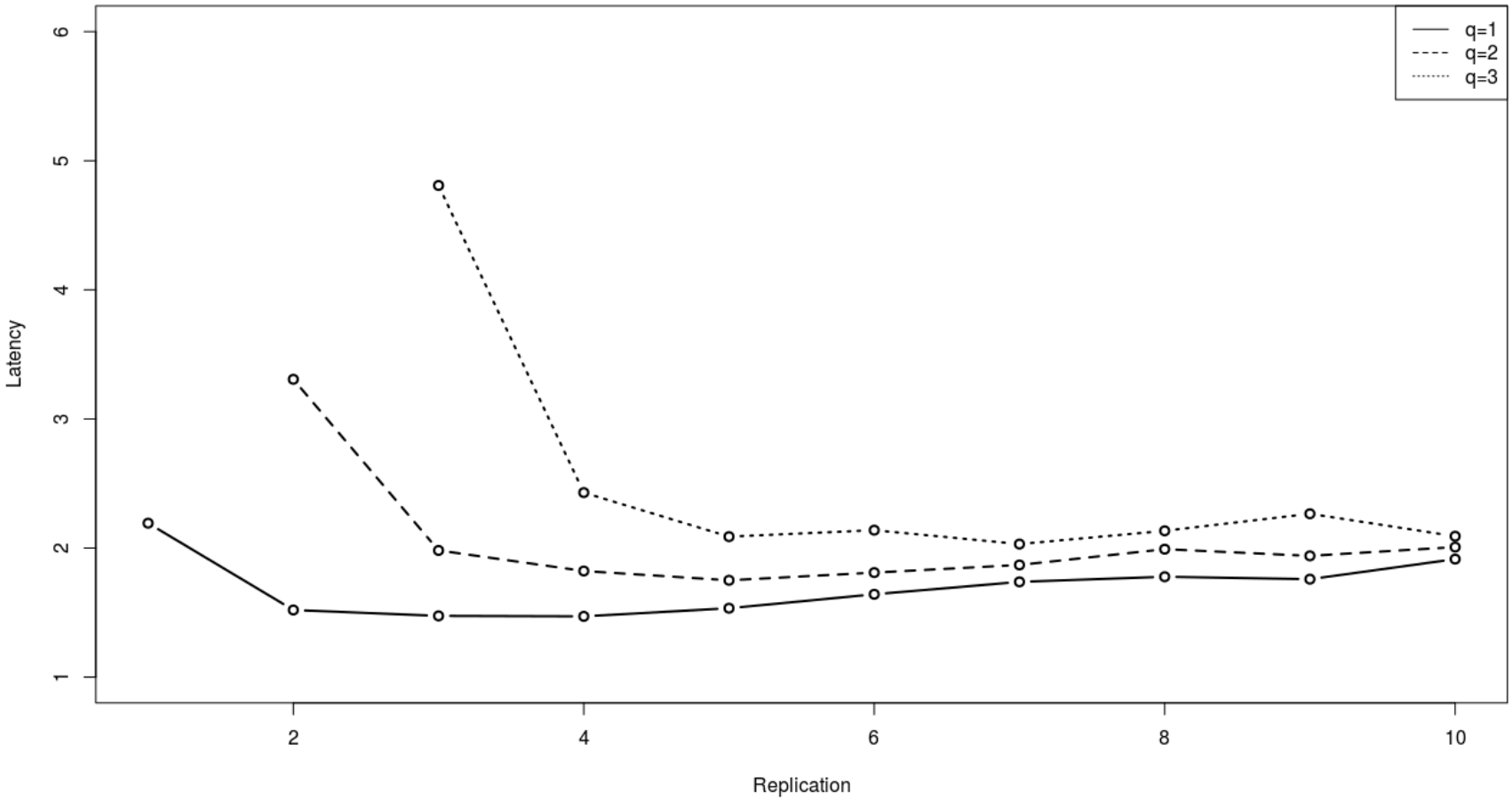
$$\frac{1}{r\mu} + \frac{C(\lfloor n/r \rfloor, \rho/r)}{\lfloor n/r \rfloor r\mu - \lambda}.$$



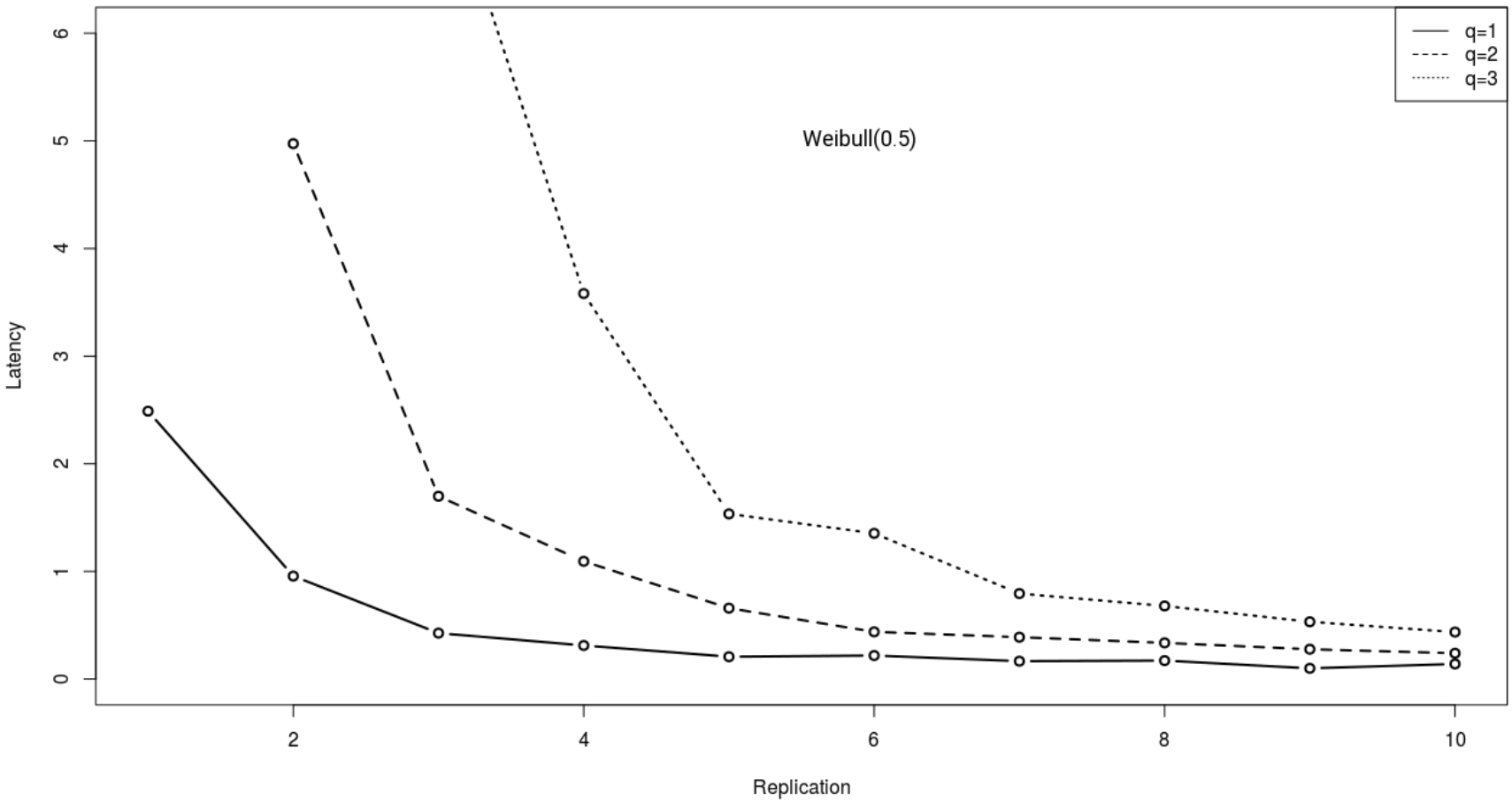
[1] Note the dependence on q

Pareto (n, r, q) replication

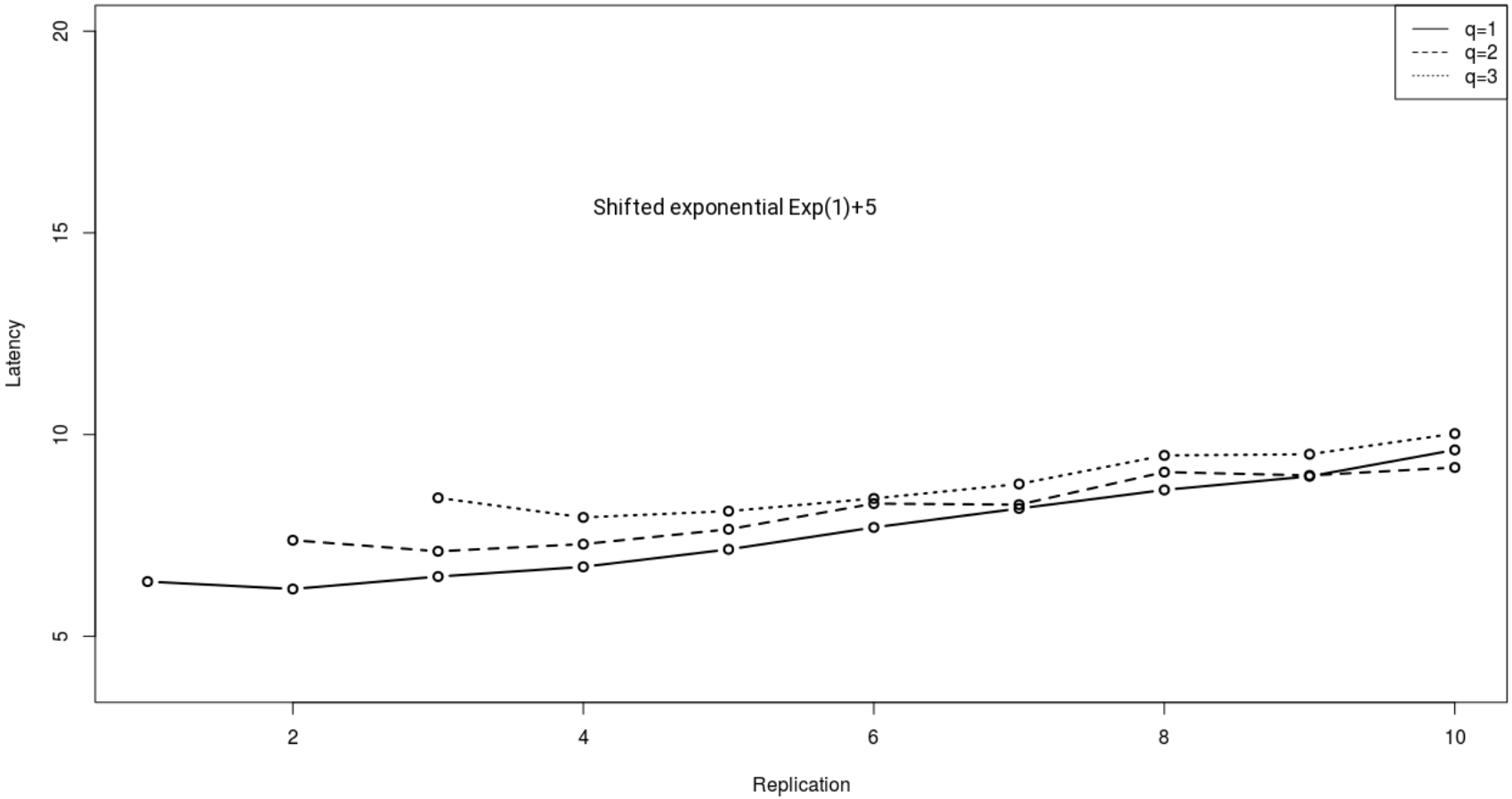
Rely on simulation. Yes, there is a tradeoff.



Weibull (n, r, q) replication



Shifted exponential (n, r, q) replication



Spoiler: Project Tails

Really it depends on the heavy-tailedness and log-concavity of service times.

... leave this for future research

Got Anything to Read?

About *Fork-Join and Split-Merge*:
G. Joshi: Efficient Redundancy Techniques to Reduce Delay in Cloud Systems
P. Fiorini, L. Lipsky: Exact analysis of some split-merge queues (based on L. Lipsky: Queueing Theory: A Linear Algebraic Approach)

About *PH distributions* and Matrix-Analytic Method:
M. Bladt, B. F. Nielsen: Matrix-Exponential Distributions in Applied Probability
P. Buchholz, J. Kriege, I. Felko: Input Modeling with Phase-Type Distributions and Markov Models
He Qi-Ming: Fundamentals of Matrix-Analytic Methods

Thank you for attention!

Rumyantsev Alexander
Institute of Applied Mathematical Research
Karelian Research Centre RAS
ar0@krc.karelia.ru

ResearcherID: L-1354-2013
ORCID: orcid.org/0000-0003-2364-5939
ScopusID: 36968331100

ResearchGate: https://www.researchgate.net/profile/Alexander_Rumyantsev