Asymptotically Optimal Load Balancing in Large-scale Heterogeneous Systems with Multiple Dispatchers

Xingyu Zhou

The Ohio State University

IFIP Performance'20

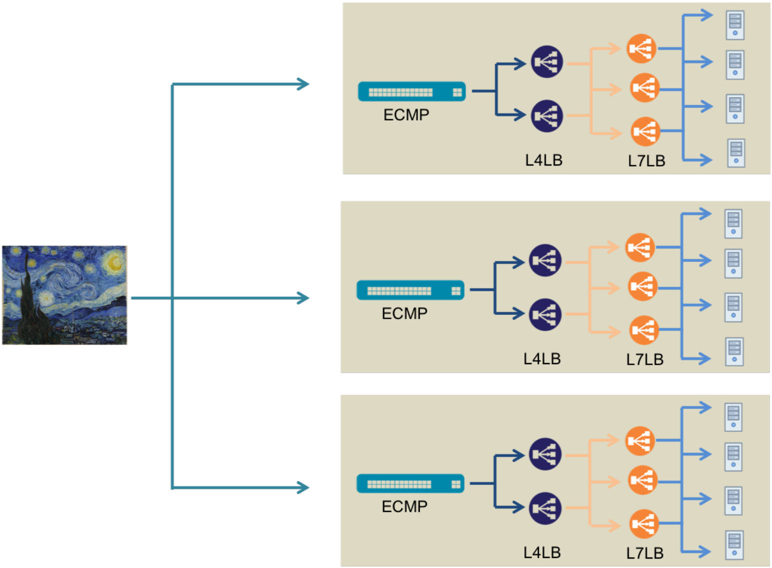
Oct. 6 2020 @ Home

Joint work with...

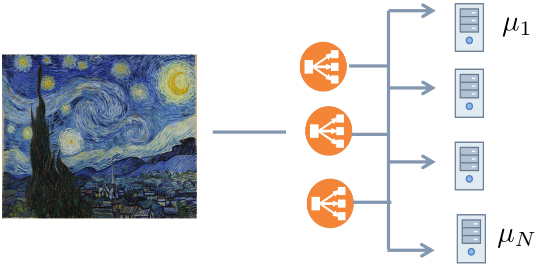
Ness Shroff, OSU

Adam Wierman, Caltech

Load Balancing...



The Building Block...



Key features:

- Multiple dispatchers
- Heterogeneous servers

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?
   - Without communication across dispatchers

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?
   - Without communication across dispatchers
3. Question: How much communication between dispatchers and servers?

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
    - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
    - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?
    - Without communication across dispatchers
3. Question: How much communication between dispatchers and servers?
    - Minimize the messages between dispatchers and servers

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?
   - Without communication across dispatchers
3. Question: How much communication between dispatchers and servers?
   - Minimize the messages between dispatchers and servers
4. Question: Can we say something about performance guarantee?

Motivating Questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly [I. Owen Garrett of NGINX]
   - Thus, how can we avoid this?
2. Question: Can each dispatcher work independently with simple implementations?
   - Without communication across dispatchers
3. Question: How much communication between dispatchers and servers?
   - Minimize the messages between dispatchers and servers
4. Question: Can we say something about performance guarantee?
   - Stability? or even delay?

Our Proposed Design Framework: LED

The Local-Estimation-Driven (LED) framework...

1. **Memory:** Each dispatcher has a local memory storing its own estimates of each server's queue length (often outdated)
2. **Dispatching:** the dispatching decision at each dispatcher is made purely based on local memory
3. **Updating:** the local memory is updated with the true queue length via messages between dispatchers and servers

Our Proposed Design Framework: LED

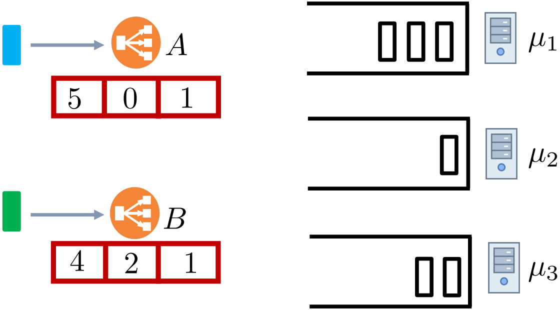
The Local-Estimation-Driven (LED) framework...

1. **Memory:** Each dispatcher has a local memory storing its own estimates of each server's queue length (often outdated)
2. **Dispatching:** the dispatching decision at each dispatcher is made purely based on local memory
3. **Updating:** the local memory is updated with the true queue length via messages between dispatchers and servers

Key contributions...

1. Sufficient conditions on dispatching and updating strategies: throughput optimality and delay optimality in heavy traffic
2. Shed light on recently proposed open problem on LB with delayed information [David Lipshutz'19]

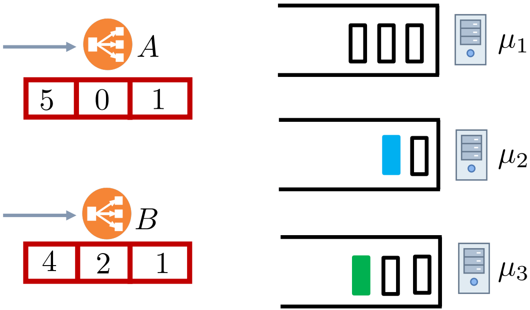
One Concrete Example...



Memory: Each dispatcher keeps its own local estimates (often outdated)...

- Dispatcher A 'believes' that: server 1 with queue length 5, server 2 with 0, and server 3 with 1
- Dispatcher B 'believes' that: server 1 with queue length 4, server 2 with 2, and server 3 with 1

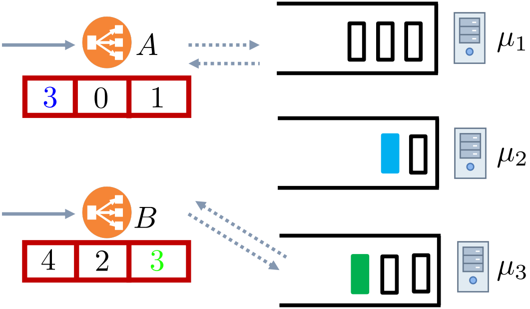
One Concrete Example...



Dispatching strategy: Local-Join-Shortest-Queue (L-JSQ)

- each dispatcher independently routes new arrivals to the server with the shortest local estimates
- e.g., Dispatcher A routes to server 2, Dispatcher B routes to server 3

One Concrete Example...



Updating strategy: Push-based update via sampling

- each dispatcher independently randomly samples d servers with probability p
- update its corresponding local estimates with the true queue lengths

Related Works...

1. LB in multiple dispatchers:
   - JIQ in [Lu et al' 11]: consider homogeneous servers; JIQ is unstable in general for fixed number of heterogeneous servers [Zhou et al' 17]
   - Pull-based algorithm in [Stolyar' 17]: heterogeneous server pools in the large-system regime; assume homogeneous loads across dispatchers

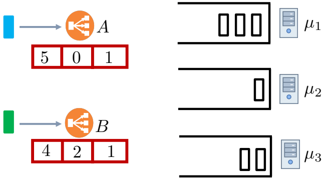
Related Works...

1. LB in multiple dispatchers:
   - JIQ in [Lu et al' 11]: consider homogeneous servers; JIQ is unstable in general for fixed number of heterogeneous servers [Zhou et al' 17]
   - Pull-based algorithm in [Stolyar' 17]: heterogeneous server pools in the large-system regime; assume homogeneous loads across dispatchers
2. LB with local memory:
   - Power-of-d in [Anselmi and Dufour' 18], JSQ in [van der Boor,' 19], a class of policies in [Gamarnik' 20]
   - All of them consider a single dispatcher

Related Works...

1. LB in multiple dispatchers:
   - JIQ in [Lu et al' 11]: consider homogeneous servers; JIQ is unstable in general for fixed number of heterogeneous servers [Zhou et al' 17]
   - Pull-based algorithm in [Stolyar' 17]: heterogeneous server pools in the large-system regime; assume homogeneous loads across dispatchers
2. LB with local memory:
   - Power-of-d in [Anselmi and Dufour' 18], JSQ in [van der Boor,' 19], a class of policies in [Gamarnik' 20]
   - All of them consider a single dispatcher
3. Most related to ours is the recent work [Vargaftik et al' 20]
   - They only consider one particular dispatching strategy, i.e., Local-JSQ.
   - They only investigate stability

Model...



- M dispatchers and N servers in discrete-time.
- **Arrival:** total number of arriving tasks $A_\Sigma(t)$ with rate λ_Σ, general distribution [1]
  - $A_\Sigma(t)$ integer-valued *i.i,d* across time-slots
  - $A_\Sigma(t) = \sum_{m=1}^{M} A^m(t)$, $A^m(t)$ arrivals at dispatcher m
  - assume $\mathbb{P}\left(A^m(t) > 0\right) \geq p_0 > 0, \quad \forall (m,t) \in \mathcal{M} \times \mathbb{N}$,
- **Service:** average number of tasks can be served at server k is μ_k, general distribution.
  - $S_n(t)$ is integer-valued, *i.i.d* across time and independent of arrival and queue lengths
- **Memory:** $\widetilde{\mathbf{Q}}^m(t)) = (Q_1^m(t), \ldots, Q_N^m(t))$
- **System states:** $Z(t) = (\mathbf{Q}(t), \{\widetilde{\mathbf{Q}}^1(t))), \ldots, \widetilde{\mathbf{Q}}^M(t)\})$

[1] with all moments bounded by absolute constants

Metrics...

In this paper, we consider both throughput optimality and delay optimality in heavy traffic...

Metrics...

In this paper, we consider both throughput optimality and delay optimality in heavy traffic... Define $\epsilon := \sum \mu_n - \lambda_\Sigma$

Definition (Throughput Optimality)

A LB policy is throughput optimal if the system is positive recurrent under any $\epsilon > 0$ and all the moments of $\|\overline{\mathbf{Q}}^{(\epsilon)}\|$ are finite

Note: this definition is stronger than simple stability

Metrics...

In this paper, we consider both throughput optimality and delay optimality in heavy traffic... Define $\epsilon := \sum \mu_n - \lambda_\Sigma$

Definition (Throughput Optimality)

A LB policy is throughput optimal if the system is positive recurrent under any $\epsilon > 0$ and all the moments of $\|\overline{\mathbf{Q}}^{(\epsilon)}\|$ are finite

Note: this definition is stronger than simple stability

Definition (Heavy-traffic Delay Optimality)

A LB policy is said to be heavy-traffic delay optimal in steady-state if the steady-state queue length vector $\overline{\mathbf{Q}}^{(\epsilon)}$ satisfies

$$\lim_{\epsilon \downarrow 0} \epsilon \mathbb{E}\left[\sum_{n=1}^{N} \overline{Q}_n^{(\epsilon)}\right] = \lim_{\epsilon \downarrow 0} \epsilon \mathbb{E}\left[\bar{q}^{\epsilon}\right],$$

where $\mathbb{E}\left[\bar{q}^{\epsilon}\right]$ is the mean queue length in resource-pooling system.

Resource-pooling system: pool all the service into one super single server

Dispatching Preference...

- Fix a dispatcher m, let $\sigma_t(\cdot)$ be a permutation of $(1, 2, \dots, N)$ that satisfies

$$\widetilde{Q}^m_{\sigma_t(1)}(t) \leq \widetilde{Q}^m_{\sigma_t(2)}(t) \leq \ldots \leq \widetilde{Q}^m_{\sigma_t(N)}(t).$$

- $P^m_n(t)$: probability of routing to server n at dispatcher m in time-slot t (again, based on local estimates)
- $\Delta^m_n(t)$: preference of the n-th shortest local estimate at dispatcher m, given by

$$\boxed{\Delta^m_n(t) := P^m_{\sigma_t(n)}(t) - \frac{\mu_{\sigma_t(n)}}{\sum \mu_n}}$$

Dispatching Preference...

- Fix a dispatcher m, let $\sigma_t(\cdot)$ be a permutation of $(1, 2, \ldots, N)$ that satisfies

$$\widetilde{Q}^m_{\sigma_t(1)}(t) \leq \widetilde{Q}^m_{\sigma_t(2)}(t) \leq \ldots \leq \widetilde{Q}^m_{\sigma_t(N)}(t).$$

- $P^m_n(t)$: probability of routing to server n at dispatcher m in time-slot t (again, based on local estimates)
- $\Delta^m_n(t)$: preference of the n-th shortest local estimate at dispatcher m, given by

$$\boxed{\Delta^m_n(t) := P^m_{\sigma_t(n)}(t) - \frac{\mu_{\sigma_t(n)}}{\sum \mu_n}}$$

  - $\Delta^m_n(t) > 0$ means that policy has stronger preference of n-th shortest local estimates compared to (weighted) random routing
  - Note that $\sum_{n=1}^N \Delta^m_n(t) = 0$
  - Key: how to allocate the zero-sum?

δ-tilted Sum Condition

$$\Delta_n^m(t) := P_{\sigma_t(n)}^m(t) - \frac{\mu_{\sigma_t(n)}}{\sum \mu_n}$$

Definition

Fix a dispatcher m, for all $1 \leq j \leq N-1$, $\sum_{n=1}^{j} \Delta_n^m(t) \geq \delta$ for some constant $\delta \geq 0$ at each time-slot t.

Intuitions: for any first k ($k < N$) shortest local estimates, it has at least δ total preference

δ-tilted Sum Condition

$$\boxed{\Delta_n^m(t) := P_{\sigma_t(n)}^m(t) - \frac{\mu_{\sigma_t(n)}}{\sum \mu_n}}$$

Definition

Fix a dispatcher m, for all $1 \le j \le N-1$, $\sum_{n=1}^{j} \Delta_n^m(t) \ge \delta$ for some constant $\delta \ge 0$ at each time-slot t.

Intuitions: for any first k ($k < N$) shortest local estimates, it has at least δ total preference

Examples: suppose all μ_n are equal and $\widetilde{\mathbf{Q}}^m(t)) = (5, 0, 1)$

- δ-tilted Sum Condition satisfied with all $\mathbf{P}^m(t)$ s.t. for some $\delta \ge 0$
- $P_2^m(t) \ge \delta + 1/3$, $P_2^m(t) + P_3^m(t) \ge \delta + 2/3$, and $\sum P_n^m(t) = 1$

Implications:

- this condition also generalizes previous definition in [Zhou et al' 17,18]
- as a result, it allows us to establish new results (e.g., L-Pod), discussed later

Main Results

We have the following sufficient condition (informal) for throughput optimality...
Define: $\mathcal{I}_n^m(t)$ indicates server n's true queue length is updated at dispatcher m

Theorem

Consider an LED policy if

- *dispatching strategy satisfies δ-tilted sum condition for some $\delta \geq 0$*
- *updating strategy satisfies that $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] > p$ for any $Z(t), m, n$ and some $p > 0$*

Then, it is throughput optimal

Remark:

- This directly generalizes LSQ policy in [Vargaftik et al' 20] in terms of stability

Main Results

We have the following sufficient condition (informal) for heavy-traffic delay optimality...

Theorem

Consider an LED policy if

- *dispatching strategy satisfies δ-tilted sum condition for some strictly positive constant δ*
- *updating strategy satisfies that $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] \geq p > 0$ for any $Z(t), m, n$ independent of previous updates;*
- *both δ and p are independent of ϵ*

Then, it is heavy-traffic delay optimal

Main Results

We have the following sufficient condition (informal) for heavy-traffic delay optimality...

Theorem

Consider an LED policy if

- *dispatching strategy satisfies δ-tilted sum condition for some strictly positive constant δ*
- *updating strategy satisfies that $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] \geq p > 0$ for any $Z(t), m, n$ independent of previous updates;*
- *both δ and p are independent of ϵ*

Then, it is heavy-traffic delay optimal

Remark:

- This directly implies a large class of LED policies are heavy-traffic delay optimal, including the specific one LSQ in [Vargaftik et al' 20]
- This also sheds light on heavy-traffic delay optimality in delayed queue length information, raised in [David Lipshutz'19]
- Moreover, the single dispatcher with accurate information is just a special case of ours

Examples of 'nice' dispatching strategies

1. L-JSQ: Local-Join-Shortest-Queue (i.e., the LSQ in [Vargaftik et al' 20])
   - choose $i^* \in \arg\min_n \{\widetilde{Q}_n^m\}$
   - $\Delta_1^m(t) = 1 - \mu_{\sigma_t(1)}/\mu_\Sigma > 0$ and all others are less than 0
   - It can be easily seen that δ-tilted sum condition is satisfied

Examples of ‘nice’ dispatching strategies

1. L-JSQ: Local-Join-Shortest-Queue (i.e., the LSQ in [Vargaftik et al' 20])
   - choose $i^* \in \arg\min_n\{\widetilde{Q}_n^m\}$
   - $\Delta_1^m(t) = 1 - \mu_{\sigma_t(1)}/\mu_\Sigma > 0$ and all others are less than 0
   - It can be easily seen that δ-tilted sum condition is satisfied
2. L-JBA: Local-Join-Below-Average
   - Let $\bar{Q}^m(t) = \frac{1}{N}\sum_n \widetilde{Q}_n^m(t)$ and $\mathcal{A} := \{n : \widetilde{Q}_n^m(t) \leq \bar{Q}^m(t)\}$
   - Then, for each $i \in \mathcal{A}$, $P_i^m(t) = \mu_i / \sum_{n\in\mathcal{A}} \mu_n$, and for $i \notin \mathcal{A}$, $P_i^m(t) = 0$.
   - it also satisfies the condition, although it needs the information on μ

Examples of 'nice' dispatching strategies

1. L-JSQ: Local-Join-Shortest-Queue (i.e., the LSQ in [Vargaftik et al' 20])
   - choose $i^* \in \arg\min_n \{\widetilde{Q}_n^m\}$
   - $\Delta_1^m(t) = 1 - \mu_{\sigma_t(1)}/\mu_\Sigma > 0$ and all others are less than 0
   - It can be easily seen that δ-tilted sum condition is satisfied
2. L-JBA: Local-Join-Below-Average
   - Let $\bar{Q}^m(t) = \frac{1}{N}\sum_n \widetilde{Q}_n^m(t)$ and $\mathcal{A} := \{n : \widetilde{Q}_n^m(t) \le \bar{Q}^m(t)\}$
   - Then, for each $i \in \mathcal{A}$, $P_i^m(t) = \mu_i / \sum_{n\in\mathcal{A}} \mu_n$, and for $i \notin \mathcal{A}$, $P_i^m(t) = 0$.
   - it also satisfies the condition, although it needs the information on μ
3. L-Pod: Local-Power-of-d
   - randomly samples d servers, join the one with the shortest local estimates
   - it turns out that even with heterogeneous servers, L-Pod can still satisfy δ-tilted sum as long as the services rates meet a certain condition

More on L-Pod

Proposition

Suppose the service rate vector $\boldsymbol{\mu} \in \mathbb{R}_+^N$ *satisfies*

$$\frac{\sum_{n=1}^{j} \mu_{[n]}}{\mu_\Sigma} + \delta \leq 1 - \frac{\binom{N-j}{d}}{\binom{N}{d}} \qquad \forall 1 \leq j \leq N-1, \tag{1}$$

for some constant $\delta \geq 0$*, in which* $\mu_{[n]}$ *is the n-th largest service rate. Then, L-Pod satisfies the* δ*-tilted sum condition.*

More on L-Pod

Proposition

Suppose the service rate vector $\boldsymbol{\mu} \in \mathbb{R}_+^N$ *satisfies*

$$\frac{\sum_{n=1}^{j} \mu_{[n]}}{\mu_\Sigma} + \delta \leq 1 - \frac{\binom{N-j}{d}}{\binom{N}{d}} \qquad \forall 1 \leq j \leq N-1, \tag{1}$$

for some constant $\delta \geq 0$*, in which* $\mu_{[n]}$ *is the n-th largest service rate. Then, L-Pod satisfies the* δ*-tilted sum condition.*

Remark:

- For the single dispatcher with accurate queue length information (which is a special case of ours), [Hurtado-Lange and Maguluri' 20]) derived similar conditions
- If $d = 1$, the only possible $\boldsymbol{\mu}$ and δ are $\mu_n = \mu$ for all n and $\delta = 0$
- If $d = N$, then all $\boldsymbol{\mu} \in \mathbb{R}_+^N$ satisfies (1) with $\delta = \mu_{min}/\mu_\Sigma > 0$

Examples of ‘nice’ updating strategies

We can generally have two categories: push-based and pull-based

Examples of 'nice' updating strategies

We can generally have two categories: push-based and pull-based

1. Push-based: each dispatcher takes the initiative to sample servers
   - e.g., at the end of each time-slot, w.p. $\tilde{p} > 0$ to randomly sample d queues and update the local estimates with the true lengths
   - thus, $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] \geq p > 0$ is satisfied with $p = \tilde{p}d/N$

Examples of 'nice' updating strategies

We can generally have two categories: push-based and pull-based

1. Push-based: each dispatcher takes the initiative to sample servers
   - e.g., at the end of each time-slot, w.p. $\tilde{p} > 0$ to randomly sample d queues and update the local estimates with the true lengths
   - thus, $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] \geq p > 0$ is satisfied with $p = \tilde{p}d/N$
2. Pull-based: each server takes the initiative to sample dispatchers
   - e.g., at the end of each time-slot, if server n finishes one or more tasks, it randomly samples one dispatcher
   - if $Q_n = 0$, it reports w.p. 1
   - if $Q_n > 0$, it reports w.p. $\tilde{p} > 0$
   - it has been verified in [Vargaftik et al' 20]), this satisfies $\mathbb{E}\left[\mathcal{I}_n^m(t) \mid Z(t)\right] \geq p > 0$ for arbitrarily small $\tilde{p} > 0$

Of course, there are many more...

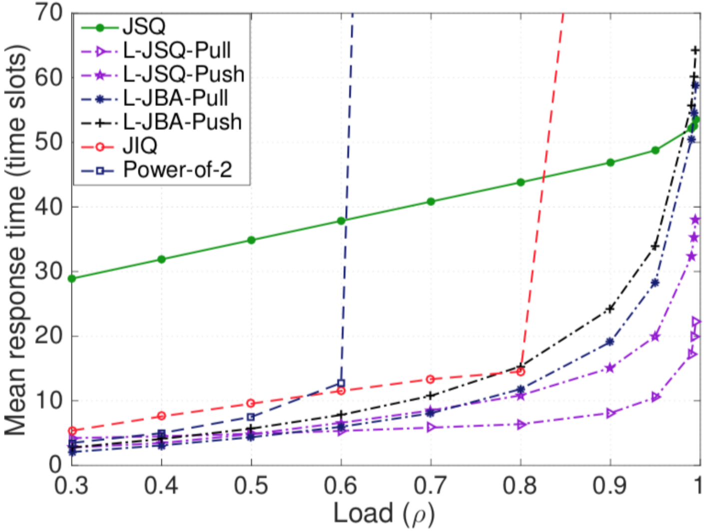
Recall our motivating questions

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beat others in performance?
   - No, due to herd behavior, it actually behaves poorly
   - Thus, how can we avoid this?

   Answer: LED could be one solution due to its intrinsic randomness

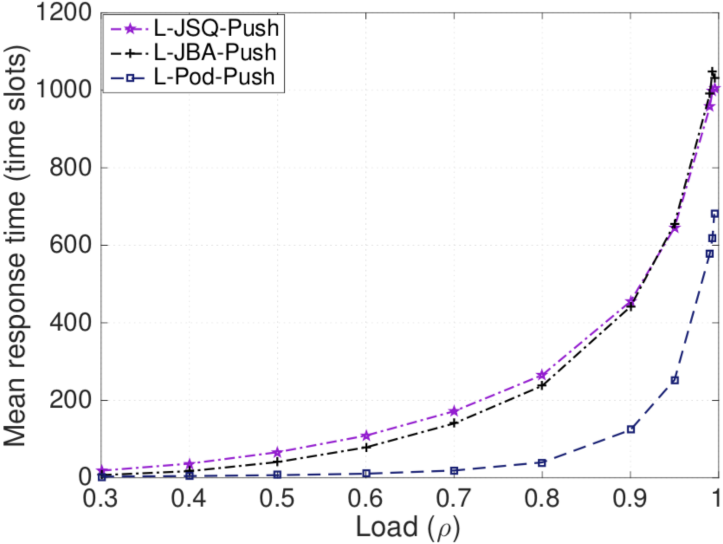
Inaccurate information helps...

100 heterogeneous servers, 10 dispatchers



Randomness further helps...

- 100 homogeneous servers, 10 dispatchers
- update probability is small $\tilde{p} = 0.01$



Recall our motivating questions

2. Question: Can each dispatcher work independently with simple implementations?

Recall our motivating questions

2. Question: Can each dispatcher work independently with simple implementations?
   - Without communication across dispatchers

   Answer: For LED, we have
   - each dispatcher totally works independently
   - immediate dispatching, i.e., no waiting for update
   - simple and fast implementations, e.g., min-heap

Recall our motivating questions

3. Question: How much communication between dispatchers and servers?

Recall our motivating questions

3. Question: How much communication between dispatchers and servers?
   - Minimize the messages between dispatchers and servers

   Answer: For LED, we have
   - the sampling and reporting probabilities can be arbitrarily small
   - of course, for practical performance, these parameters can be tuned to trade-off between messages and performance

Recall our motivating questions

4. Question: Can we say something about performance guarantee?

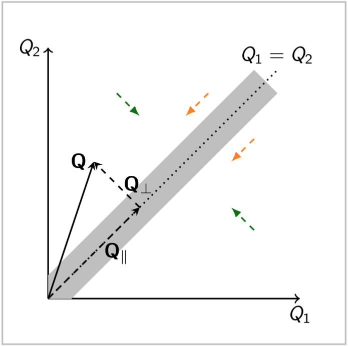
Recall our motivating questions

4. Question: Can we say something about performance guarantee?
   - Stability? or even delay?

   Answer: For LED, we have
   - throughput optimality
   - delay optimality in heavy traffic

Main ideas behind proofs

- The main techniques are based on drift-based [Eryilmaz and Srikant'12])
- In particular, we utilize the sufficient conditions for throughput and heavy-traffic optimality in [Zhou et al'17], illustrated as follows



- Throughput optimality needs positive drift, obtained via

$$\sum_{t=1}^{T} \mathbb{E}\left[\langle \mathbf{Q}, \mathbf{A}-\mathbf{S}\rangle \mid \mathbf{Q}\right] \approx -\epsilon \|\mathbf{Q}\|$$

- Heavy-traffic optimality needs positive drift, obtained via

$$\sum_{t=1}^{T} \mathbb{E}\left[\langle \mathbf{Q}_\perp, \mathbf{A}-\mathbf{S}\rangle \mid \mathbf{Q}\right] \approx -\delta \|\mathbf{Q}_\perp\|$$

Main ideas behind proofs

Three additional challenges arise in our settings...

1. A more general dispatching condition (i.e., δ-tilted sum condition)
   - it exists even when the queue lengths are accurate
   - we draw inspirations from [Hurtado-Lange and Maguluri' 20]) to have a nice bound on the inner product between $\mathbf{Q}$ and $\mathbf{A}$
2. Outdated queue lengths information
   - our strategy is to do a decomposition
   - first, establish necessary drifts via dispatching strategy, assuming the queue lengths are accurate
   - second, bounding the error via update condition
3. System state includes local estimates
   - hence, for throughput optimality, they should also be bounded

Conclusion...

The LED combined with sufficient conditions give affirmative answers to all key questions...

1. Question: With multiple dispatchers, does Join-Shortest-Queue still beats others in performance?
   Answer: LED could be one solution due to its intrinsic randomness
2. Question: Can each dispatcher work independently with simple implementations?
   Answer: LED achieve independence, easy implementations
3. Question: How much communication between dispatchers and servers?
   Answer: LED, has the flexibility to tune the probability $\tilde{p}$
4. Question: Can we say something about performance guarantee?
   Answer: LED, can be throughput optimal and delay optimal in heavy traffic

Future Works

There are several interesting directions for LED...

1. Beyond the traditional heavy-traffic regime?
   - As pointed out by [Zhou et al' 18]), heavy-traffic delay optimal is a coarse metric in certain sense
   - How about waiting probability in large-system regimes?
2. How about continuous-time systems?
3. How about LED on graphs?
   - each node can serve a job or dispatches to neighbors
   - each node keeps local estimates of its neighbors
   - purely based on local memory to dispatch
   - infrequent update via communications between nodes

Thank you!

Q & A

Throughput optimality...

1. We consider the Lyapunov function
   $W(Z(t)) = \|\mathbf{Q}(t)\|^2 + \sum_{m=1}^{M} \left\| \mathbf{Q}(t) - \widetilde{\mathbf{Q}}^m(t) \right\|_1$
2. Let $X_n^m(t) \triangleq |Q_n(t) - \widetilde{Q}_n^m(t)|$, the drift is

$$D(Z(t_0)) = D_Q(t_0) + \sum_{m=1}^{M} \sum_{n=1}^{N} D_{X_n^m}(t_0) \tag{2}$$

   where

$$D_Q(t_0) \triangleq \mathbb{E}\left[\|\mathbf{Q}(t_0+T)\|^2 - \|\mathbf{Q}(t_0)\|^2 \mid Z(t_0) \right]$$
$$D_{X_n^m}(t_0) \triangleq \mathbb{E}\left[X_n^m(t_0+T) - X_n^m(t_0) \mid Z(t_0) \right]$$

3. $D_{X_n^m}(t_0) \leq -pX_n^m(t) + 2T\mu_\Sigma$

Throughput optimality (Cont'd)

4. Turn to $D_Q(t_0) \triangleq \mathbb{E}\left[\|\mathbf{Q}(t_0+T)\|^2 - \|\mathbf{Q}(t_0)\|^2 \mid Z(t_0)\right] \approx \sum \mathbb{E}\left[\langle \mathbf{Q}, \mathbf{A} - \mathbf{S} \rangle \mid Z(t_0)\right] + K$
5. We can decompose the first term into $(\beta_n^m(t) := P_n^m(t) - \mu_n/\mu_\Sigma)$

$$\text{RHS} \approx \underbrace{\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M}\left(Q_n(t) - \widetilde{Q}_n^m(t)\right)\beta_n^m(t)\lambda_m \mid Z\right]}_{\mathcal{T}_1}$$

$$+ \underbrace{\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M}\widetilde{Q}_n^m(t)\beta_n^m(t)\lambda_m \mid Z\right]}_{\mathcal{T}_2} - \frac{\epsilon\mu_{min}}{\mu_\Sigma}\|\mathbf{Q}(t_0)\|_1 .$$

6. For $\mathcal{T}_1$, by update condition, we have a constant bound on it

Throughput optimality (Cont'd)

7. Turn to $\mathcal{T}_2 = \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M} \widetilde{Q}_n^m(t)\beta_n^m(t)\lambda_m \mid Z\right]$
8. It is equal to $\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M} \widetilde{Q}_{\sigma_t(n)}^m(t)\Delta_n^m(t)\lambda_m \mid Z\right]$
9. The green term can be written as

$$\sum_{m=1}^{M}\left(\widetilde{Q}_{\sigma_t(1)}^m(t)\sum_{n=1}^{N}\Delta_n^m(t)\right) \tag{3}$$

$$+\sum_{m=1}^{M}\left(\sum_{k=2}^{N}\left(\sum_{n=k}^{N}\Delta_n^m(t)\right)\left(\widetilde{Q}_{\sigma_t(k)}^m(t)-\widetilde{Q}_{\sigma_t(k-1)}^m(t)\right)\right) \tag{4}$$

10. (3) is zero as $\sum_{n=1}^{N}\Delta_n^m(t) = 0$
11. (4) less than zero since $\sum_{n=k}^{N}\Delta_n^m(t) \le -\delta$ by δ-tilted sum condition

Heavy-traffic delay optimality...

1. We wish to establish $\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\langle \mathbf{Q}_\perp, \mathbf{A}-\mathbf{S}\rangle \mid Z\right] \approx -\delta' \|\mathbf{Q}_\perp\|$, δ' independent of ϵ
2. The key term $\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\langle \mathbf{Q}_\perp, \mathbf{A}\rangle \mid Z\right]$ can be written as

$$
\begin{aligned}
&\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N} Q_{\perp,n}(t) \sum_{m=1}^{M} \beta_n^m(t)\lambda_m \mid Z\right] \\
=&\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M} \left(\widetilde{Q}_n^m(t) - \bar{Q}^m(t)\right) \beta_n^m(t)\lambda_m \mid Z\right] \quad (5)\\
&+\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M} \left(Q_n(t) - \widetilde{Q}_n^m(t)\right) \beta_n^m(t)\lambda_m \mid Z\right] \quad (6)\\
&+\sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{n=1}^{N}\sum_{m=1}^{M} \left(\bar{Q}^m(t) - Q_{\text{avg}}(t)\right) \beta_n^m(t)\lambda_m \mid Z\right]. \quad (7)
\end{aligned}
$$

   where $\bar{Q}^m(t) := \frac{1}{N}\sum_n \widetilde{Q}_n^m(t)$ and $Q_{\text{avg}} := \frac{1}{N}\sum_n Q_n(t)$
3. By updating condition, (6) and (7) both can be upper bounded (properly chosen T)

Heavy-traffic delay optimality (Cont'd)

4. Turn to the green term, it can be written as

$$(5) = \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{m=1}^{M}\sum_{n=1}^{N} \widetilde{Q}^m_{\sigma_t(n)}(t)\Delta^m_n(t)\lambda_m \mid Z\right]$$

5. Follow the same decompositions as in (3) and (4), we have

$$(5) \le -\delta \sum_{t=t_0}^{t_0+T-1} \mathbb{E}\left[\sum_{m=1}^{M}\sum_{n=1}^{N} \widetilde{Q}^m_{max}(t) - \widetilde{Q}^m_{min}(t)\lambda_m \mid Z\right]$$

6. By a careful sample-path analysis, we have for some constant K

$$\begin{aligned}(5) &\le -\delta f(p)\lambda_{min}\left(Q^m_{max}(t_0) - Q_{min}(t_0)\right) + K \\ &\le -\delta' \left\|\mathbf{Q}_\perp(t_0)\right\| + K\end{aligned}$$