An analysis of sparse, limited flexibility, service architectures

John N. Tsitsiklis, Kuang Xu

July 2015
The general theme – some flexibility goes a long way

- Supply chain management
  - Jordan & Graves, 1995
  - Simchi-Levi & Wei, 2012
  - Chen et al., 2014, ...

- Supermarket model & extensions (load balancing/distribution)
  - Mitzenmacher, 1996
  - Vvedenskaya et al., 1996
  - Stolyar, 2015, ...

- Content provision
  - Leconte et al., 2012
  - Shah & deVeciana, 2015
Outline

- Service architectures with multiple job streams and servers
- Server flexibility: first example (fraction of capacity in global server)
- Sparsity requirement
- Second example: modular architecture
- Main model: based on expander graphs
  - scheduling policy and its analysis
- Concluding remarks
A space of service system architectures

**Context**
- server farms
- content provision
- call centers

**Stream** $i$
- type $i$ jobs
- demand for video $i$
- question on subject $i$

**No flexibility**
- $\lambda \rightarrow \begin{array}{c}
\text{local queues} \rightarrow 1
\end{array}$
- delay $\sim 1/(1 - \lambda)$

**Full flexibility**
- $\lambda \rightarrow \begin{array}{c}
\text{global (flexible server)} \rightarrow 1
\end{array}$
- delay $\rightarrow 0$

- Consider intermediate ("limited flexibility") architectures
A partially flexible system

- \( N \) independent Poisson streams, \( \lambda < 1 \)
- local exponential servers, rate \( 1 - p \)
- global (flexible server), rate \( pN \)
- \( Q_i^{(N)} \): # of jobs in queue \( i \), in steady-state
- \( p = 0 \): \( Q_i^{(N)} \) is geometric, mean \( \frac{1}{1 - \lambda} \)
- \( p = 1 \): \( Q_i^{(N)} \) \( \Rightarrow 0 \), as \( N \to \infty \)
Old results [JNT+KX, 2012]

\[ P \left( Q_i^{(N)} \geq i^* \right) \rightarrow 0 \]

\[ i^* = \log \frac{1}{1-p} \frac{1}{1 - \lambda} \]

- \( \mathbb{E}[\text{delay}] = O\left( \log \frac{1}{1 - \lambda} \right) \)

![Graph showing average queue length vs. traffic intensity with curves for p=0.05 and p=0, highlighting no pooling and a little pooling cases.](image)
Back to the original motivation

No flexibility
degree = 1

delay \sim 1/(1 - \lambda)

Limited flexibility
degree \ll n

Full flexibility
degree = n

delay \rightarrow 0

• Every node has one skill vs. all skills
• Previous model \approx fraction \ p \ of servers have all skills
• Every node has \ d \ll n \ skills
• Can we get delay \rightarrow 0? Architecture/graph? Scheduling policy?
One solution: Modular architecture

- All arrival rates are equal and $< 1$: delay $\to 0$
- But! not robust w.r.t. non-uniform arrival rates
  - e.g., if half of the streams have $\lambda_i = 0$
  and half have $\lambda_i \approx 2$

- fully connected clusters
  - $d$ queues and $d$ servers
- $d \to \infty$, e.g., $d \approx \sqrt{n}$
Randomized modular architecture

- Suppose that: \( \sum_i \lambda_i \leq \rho n \)  \( \rho < 1 \): load factor
  \[ \lambda_i \leq u \]  \( u \): fluctuation parameter  \( u \ll d \)

- Create \( d \)-stream clusters at random

- For any allowed rate vector, \( P(\text{stable and delay } \ll 1) \to 1 \)
  - But! Bad rate vectors will also exist
Robustness requires expanders

- Want to be able to handle “allowed” arrival vectors with:
  \[ \lambda_i \leq u < d \]
  \[ \sum_i \lambda_i \leq \rho n \quad \rho < 1 \]

- A set \( S \) of streams must be connected to at least \( u \cdot |S| \) servers
  (when \( |S| \leq \rho n / u \))
  \( \Rightarrow \) need an expander graph

- **Theorem: (Robustness; Large Capacity Region)**
  Expander with expansion parameter \( \approx u \)
  \( \Rightarrow \) all allowed \( (\lambda_i) \) are feasible
  (corollary of max-flow/min-cut theorem)
Random graphs $\rightarrow$ expanders

- **Fact:** If $d = (1 + \epsilon)u$, a random graph with expected degree $d$ has the desired expansion property, with high probability.

- How about delay?

- **Theorem [JNT+KX, 2013]:** Can design a policy such that: For allowed $(\lambda_i)$: delay $\ll 1$, with high probability

- But maybe, for any given graph, some “allowed” $(\lambda_i)$ will have “bad” delay?

- **Theorem [JNT+KX, 2015]:** Assume expander, $u \leq cd$. Can design a policy such that: For all allowed $(\lambda_i)$: delay $\ll 1$. 

- Theorem [JNT+KX, 2013]: Can design a policy such that: For allowed $(\lambda_i)$: delay $\ll 1$, with high probability
A virtual queue policy

batch the arrivals: $\rho b$ jobs \quad time \approx \frac{b}{n}

service intervals: $s = (\rho + \epsilon)\frac{b}{n} < \frac{b}{n}$

- At start of service interval:
  - If “free”: make all servers busy
  - at end: $\approx (\rho + \epsilon)b$ free servers
  - assign all $\rho b$ jobs to servers, if possible
  - success: batch departs, next service interval is free
  - failure: do something simple, until all jobs are assigned

$u \leq cd < d$

$log n \ll d \ll b \ll n$
A virtual queue policy

batch the arrivals: $\rho b$ jobs $\quad$ time $\approx b/n$

$u \leq cd < d$

$\log n \ll d \ll b \ll n$

service intervals: $s = (\rho + \epsilon) \frac{b}{n} < \frac{b}{n}$

- Service time of a batch: $s$ (one service slot)
  - plus more service slots in case of failure (probability $q$)
  - additional time: mean $O(b)$; variance: $O(b^2)$
- Key lemma: $q = O(1/n^2)$
- $\#$ batches: G/G/1 queue $\quad$ $\mathbb{E}[\text{svc. time}] \approx s < b/n$ Stable!
- Kingman’s formula: time of a batch in virtual queue $= O(s)$
The key lemma: $P(“failure to assign”) \text{ is small}$

batch the arrivals: $\rho b$ jobs \hspace{1cm} time $\approx b/n$

\begin{itemize}
  \item batch: $\rho b$ jobs arrive; to $\leq \rho b$ queues \hspace{1cm} keep $\approx \rho b$ “left nodes”
  \item during service interval for batch:
    \hspace{0.5cm} about $(\rho + \epsilon)b$ free servers ("right nodes"), w.h.p.
  \item Initial graph is expander: set $S$ has (about) $\geq d|S|$ neighbors
  \item Random subgraph is expander, w.h.p., expansion $\approx d(b/n)$
  \item each left node has “supply” (about) $\leq u(b/n)$, w.h.p.
    \hspace{0.5cm} $\Rightarrow$ exists feasible assignment (w.h.p.)
\end{itemize}

$u \leq cd < d$

$log n \ll d \ll b \ll n$
The key sub-lemma: random subsets of expanders are expanders

- Initial graph is expander: set $S$ has (about) $\geq d|S|$ neighbors
- In subgraph:
  a neighbor is still there, with prob. $\approx b/n$
  set $S$ has $\geq d|S|b/n$ neighbors, w.h.p.
  (delicate) union bound: all sets $S$ have $\geq d|S|b/n$ neighbors, w.h.p.
  subgraph is expander, expansion $d(b/n)$ (w.h.p.)
Recapitulation

batch the arrivals: $\rho b$ jobs  

time $\approx b/n$

service intervals:  

$\begin{align*} s &= (\rho + \epsilon) \frac{b}{n} < \frac{b}{n} \

\end{align*}$


- Choose an expander graph  

expansion parameter, $d > u$, fluctuation parameter

- Allowed flows ($\lambda_i < u$) are feasible

- Batch & match $\rho b$ jobs
  
  - succeed with high probability

  - delay of job: $\approx$ batching time $\approx b/n$  

    plus “service time of batch” $\approx s \approx b/n$
Comments

- Policy uses centralized information
collect batch of jobs and free servers and assign
- Can design architecture/policy with randomized routing
that does not use global state information
  - clusters as in random modular architecture
  - connect clusters according to sparse expander
- But policy needs to know the $\lambda_i$
- Open question:
  architecture+policy with vanishing delay and low informational requirements?
Back to the big picture

• Small degree of centralization (or resource pooling, or flexibility) can yield significant benefits.
  – architecture/policy design may be nontrivial

• Technical extensions:
  – relax Poisson/exponential assumptions
  – heavy-tailed service times
  – time-varying $\lambda_i$
  – multiple-layer models

• Models that incorporate specific application aspects
  – Call centers
  – Supply chains
  – Decision making systems
  – …
Thank you!