SNC-Meister: Admitting More Tenants With Tail Latency SLOs

Timothy Zhu
Carnegie Mellon University

Daniel S. Berger
University of Kaiserslautern

Mor Harchol-Balter
Carnegie Mellon University

Presented By: Zane Ma & Shuo Feng
Cloud Request Latency

High performance cloud computing in a single datacenter

Ex: MapReduce, Heron, HDFS

Cloud networks provide latency service-level objectives (SLOs)

Typically guarantee 99% or 99.9% request latency, rather than packet latency
Goal: Achieving high tenancy while meeting tail latency SLOs

High performance cloud computing in a single datacenter

Ex: MapReduce, Heron, HDFS

Cloud networks provide latency service-level objectives (SLOs)

Typically guarantee 99% or 99.9% request latency, rather than packet latency
Latency Causes

Assumption: typical behavior, no hardware failure, flash crowds, etc.

**Short lived bursts** caused by network queues and services
Modeling Latency

Deterministic Network Calculus

Calculate **fixed maximum** rate/burst constraints from historical traces

Consider worst case scenario from **adversarial coordination** (i.e. 100% latency)

Used by Silo (SIGCOMM 2015), QJump (NSDI 2015), PriorityMeister (SoCC 2014)
Modeling Latency

Deterministic Network Calculus

Calculate **fixed maximum** rate/burst constraints from historical traces

Consider worst case scenario from **adversarial coordination** (i.e. 100% latency)

Used by Silo (SIGCOMM 2015), QJump (NSDI 2015), PriorityMeister (SoCC 2014)

Stochastic Network Calculus

Model maximum rate/ burstiness as a **probabilistic distribution**

Does not assume all tenants are adversarially correlated - **lower target latency percentile** (e.g. 99.9%)
Modeling Latency

Deterministic Network Calculus

Stochastic Network Calculus

- peak burst rate
- peak burst rate

\[ P(X=k) \]

\[ \lambda = 1, \lambda = 4, \lambda = 10 \]
Modeling Latency

Deterministic Network Calculus

Stochastic Network Calculus

99.9% latency
Modeling Latency

75% more admitted

9.9% latency
SNC Example
SNC Example

Arrival Processes

Tenant VM 1

Tenant VM 2

Queue

Switch

Queue

Server VM

A1

A2

A3
SNC Example

Arrival Processes

Queue

Switch

Queue

Service Processes

Tenant VM 1

Tenant VM 2

A1

A2

S1

S2

A3

Server VM
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \textbf{switch} latency + \textbf{server} latency
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = Latency(A1, S1, 0.99) + Latency(A3, S2, 0.99)
Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \text{Latency}(A1, S1, 0.99) + \text{Latency}(A3, S2, 0.99)

\textbf{S1} slowed down by \textbf{A2}!
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \text{Latency}(A1, S1, 0.99) + \text{Latency}(A3, S2, 0.99)

\textbf{S1} slowed down by \textbf{A2}! \implies S'1 = \text{Leftover}(S1, A2)
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = Latency(A1, S1 S'1, 0.99) + Latency(A3, S2, 0.99)
S'1 = Leftover(S1, A2)
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \text{Latency}(A1, S'1, 0.99) + \text{Latency}(A3, S2, 0.99)

S'1 = \text{Leftover}(S1, A2)

A3 = \text{Output}(A1, S'1)
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \text{Latency}(A1, S'1, 0.99) + \text{Latency}(A3, S2, 0.99)
S'1 = \text{Leftover}(S1, A2)
A3 = \text{Output}(A1, S'1)
Adding latencies does not preserve SLO %!
SNC Example

Goal: Get 99% latency SLO bound between Tenant VM 1 and Server VM

Total latency = \text{Latency}(A1, S'1, 0.99) + \text{Latency}(A3, S2, 0.99)
S'1 = \text{Leftover}(S1, A2)
A3 = \text{Output}(A1, S'1)
Adding latencies does not preserve SLO %! \text{Convolution}(L1, L2, 0.99)
## SNC Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency(A, S, N)</td>
<td>$N%$ latency for a given A, S</td>
</tr>
<tr>
<td>Leftover(S, A)</td>
<td>S adjusted/reduced by A</td>
</tr>
<tr>
<td>Output(A, S)</td>
<td>Resultant output distribution of A and S</td>
</tr>
<tr>
<td>Convolution(L1, L2)</td>
<td>Combine latencies L1, L2</td>
</tr>
<tr>
<td>Aggregation(A1, A2)</td>
<td>Multiplexed A1 and A2</td>
</tr>
</tbody>
</table>
SNC Implementation Challenges

- SNC order of operations optimizations
- Tunable dependencies between tenants
- Modeling burstiness - Markov Modulated Poisson Process
- Programming language abstraction for applying SNC operators
SNC Implementation Challenges

- SNC order of operations optimizations
- Tunable dependencies between tenants
- Modeling burstiness - Markov Modulated Poisson Process
- Programming language abstraction for applying SNC operators
SNC Implementation Challenges

- SNC order of operations optimizations
- Tunable dependencies between tenants
- Modeling burstiness - Markov
- Modulated Poisson Process
- Programming language abstraction for applying SNC operators

Switching between high and low phases

\[ Q = \begin{pmatrix} p_{hh} & p_{hl} \\ p_{lh} & p_{ll} \end{pmatrix} \]
SNC Implementation Challenges

SNC order of operations optimizations

Tunable dependencies between tenants

Modeling burstiness - Markov Modulated Poisson Process

Programming language abstraction for applying SNC operators

```
ArrivalProcess* A1 = new MMPP(traceT1);
ArrivalProcess* A2 = new MMPP(traceT2);
ServiceProcess* S1 = new NetworkLink(bandwidth);
ServiceProcess* S2 = new NetworkLink(bandwidth);

ServiceProcess* S1x2_A2 = new Leftover(S1x2, A2);
ServiceProcess* S1x2_A2 = new Leftover(S1x2, A2);
ServiceProcess* S1x2_A2 = new Leftover(S1x2, A2);
```
Experimental Setup

Silo: DNC, fixed 1.5Kb bursts, trial and error manual bandwidth selection

Silo++: Silo with dynamic bandwidth selection

QJump: manual priority class assignment

QJump++: QJump with automatically assigned priority class

PriorityMeister: automatically derived rates from tenant trace

Real production 2015 traces from large internet company
Results

More Tenants

High Network Utilization
Results

Scales to high SLO %

#Tenants Scales to Cluster Size

![Graph showing latency percentile vs. number of 9s and tenants admitted vs. scaling factor.](image)
Future Work / Discussion

Bootstrapping representative historical traces/logs is a chicken-and-egg problem. How can we improve the process?

How can we build fault-tolerance into SNC-Meister? Any practical SLO mechanism should account for as many failure scenarios as possible.

The paper makes an assumption about latency within a single datacenter, why do we need this assumption? What if this assumption is not met?

When most of the tenants are dependent on one another, why does SNC show higher latency than DNC?
Backup Slides
## SNC Operators

<table>
<thead>
<tr>
<th>Purpose</th>
<th>$\rho(\cdot)$</th>
<th>$\sigma(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival process $A$ for MMPP with transition matrix $Q$ and diagonal matrix $E(\theta)$ of each state’s MGF</td>
<td>$\rho_A(\theta) = sp(E(\theta)Q)$</td>
<td>$\sigma_A(\theta) = 0$</td>
</tr>
<tr>
<td>Service process $S$ for network link with bandwidth $R$</td>
<td>$\rho_S(\theta) = -R$</td>
<td>$\sigma_S(\theta) = 0$</td>
</tr>
<tr>
<td>Leftover operator $\ominus$ for service process $S$ and arrival process $A$</td>
<td>$\rho_{S\ominus A}(\theta) = \rho_A(\theta) + \rho_S(\theta)$</td>
<td>$\sigma_{S\ominus A}(\theta) = \sigma_A(\theta) + \sigma_S(\theta)$</td>
</tr>
<tr>
<td>Output operator $\odot$ for service process $S$ and arrival process $A$</td>
<td>$\rho_{A \odot S}(\theta) = \rho_A(\theta)$</td>
<td>$\sigma_{A \odot S}(\theta) = \sigma_A(\theta) + \sigma_S(\theta) - \frac{1}{\theta} \log \left(1 - e^{\theta(\rho_A(\theta) + \rho_S(\theta))}\right)$</td>
</tr>
<tr>
<td>Aggregate operator $\oplus$ for arrival process $A_1$ and arrival process $A_2$</td>
<td>$\rho_{A_1 \oplus A_2}(\theta) = \rho_{A_1}(\theta) + \rho_{A_2}(\theta)$</td>
<td>$\sigma_{A_1 \oplus A_2}(\theta) = \sigma_{A_1}(\theta) + \sigma_{A_2}(\theta)$</td>
</tr>
<tr>
<td>Convolution operator $\otimes$ for service process $S_1$ and service process $S_2$</td>
<td>$\rho_{S_1 \otimes S_2}(\theta) = \max{\rho_{S_1}(\theta), \rho_{S_2}(\theta)}$</td>
<td>$\sigma_{S_1 \otimes S_2}(\theta) = \sigma_{S_1}(\theta) + \sigma_{S_2}(\theta) - \frac{1}{\theta} \log \left(1 - e^{-\theta \rho_{S_1}(\theta) - \rho_{S_2}(\theta)}\right)$</td>
</tr>
<tr>
<td>Tail latency $L$ for percentile $p$, arrival process $A$, and service process $S$</td>
<td>$L = \min_{\theta} \frac{1}{\rho_S(\theta)} \log \left((1 - p) \ast (1 - \exp(\theta \ast (\rho_A(\theta) + \rho_S(\theta))))\right) - \frac{1}{\rho_S(\theta)}(\sigma_A(\theta) + \sigma_S(\theta))$</td>
<td></td>
</tr>
</tbody>
</table>