Tackling Latency via Replication in Distributed Systems

Zhan Qiu, Imperial College London
Juan F. Pérez, University of Melbourne
Peter G. Harrison, Imperial College London

Failure types

Request failure:

• Request in service is lost
• Server is not affected
• Communication failures
• Timeouts of resources with limited availability
• Outputs failing to meet time constraints

Impact:

• Poor service quality
• Economic losses, environmental damage
State-of-the-Art Strategies

Fault-tolerance mechanisms

- **Retry**: typically after a timeout handled by a central scheduler
  
  \[\Rightarrow\text{ Introduces unacceptable delay!}\]

- **Attack of the clones**:  
  
  - Launch multiple *clones* of a request  
  - Use the *first* successful result returned  
  - **Cancel** all outstanding replicas.
Motivation: Low Utilization

- **Low utilization**: heavy concurrent replication is appealing in the light of the low utilization common in data centers.

- **Example**: Facebook traces reveal median CPU and memory utilization under 20%.
Motivation: Cost-efficiency

- Much of the energy consumption is wasted at low utilization.
- An idle server consumes 65% of its peak power consumption.

*Cost-effective to use these idling resources for running extra replicas of requests.*
Motivation: It Works!

• Efficient to improve the system reliability.
• Has the potential to reduce response times
• Overall latency: minimum of the delays of all the replicas.
Open Questions

• **When** is the reduction in latency realized?
• Under what **conditions**?
• How **large** is the potential reduction?
• How many clones to have?
• **Centralized** set-up or **distributed**?
System Set-up

Each node

- Composed of $r$ servers
- Serving a job of replication level $r$
System Set-up

- **Centralized** set-up
- **Distributed** set-up

- Request arrivals: Markovian Arrival Process (MAP)
- **Replica** time-to-failure: exponentially distributed
- **Replica** processing times: exponentially distributed
- Phase type **request** response time
Challenges

- Mean response time? Response time distribution
- System with replication: no standard model

- Analyzing distributed set-up is more challenging
  - Synchronized arrivals correlates all the queues
  - Individual replicas fail asynchronously
Steps

**Target:**

The job response time distribution

**Steps:**

1. The waiting-time distribution
2. The service-time distribution
The Centralized Set-up

- **Without** replication
- **With** one extra replica
The Centralized Set-up

• *With* one extra replica

**Service state:**

• $L_i(t)$: number of tasks with $i$ replicas in service at time $t$

• $Y(t) = (i,j)$: state of the youngest job in service
  • $i$ replicas of the youngest job are in service
  • $j$ replicas are waiting in the queue
  • $r-i-j$ replicas already failed

$Y(t) = (1,1)$
$L1(t) = 1$
$L2(t) = 1$
The Distributed Set-up

Challenge: the queue-length

Solution:
1. Sort the queues by their lengths
2. Focus on the queue length difference.
3. Limit C: maximum queue-length difference.
The Distributed Set-up

**Challenge:**
Dependence between waiting and service processes

**Solution:**
- *Look backwards in time!*
- Consider jobs that start service with and without waiting separately.
Approximation errors

Approximation errors compared with simulation results

• Example: $r = 3$, 90% reliability, 95th percentile

![Diagram showing approximation errors for Poisson and MAP arrivals with different load levels and C values.](image_url)
Approximation errors

Approximation errors compared with simulation results

- Example: \( r = 3, \) 10% reliability, 95th percentile

![Graphs showing approximation error comparison for Poisson and MAP arrivals with different loads and rates.](image-url)
The Effect of Replication

Example: Poisson arrivals, 90% reliability
The Effect of the Reliability

Example: Poisson arrivals

90% reliability

50% reliability
The Effect of the Arrival Process

Example: 90% reliability
The Effect across the Distribution

Example: Poisson arrivals, 90% NR-reliability, 0.3 load
Distributed vs Centralized

1. **Advantage of the distributed set-up:**
   - More flexibility.
   - Always spreads tasks across different servers.

2. **Response times**: centralized set-up achieves lower ones
   - How much?
Distributed vs Centralized

- Example: Poisson arrival, \( r = 2 \), 0.3 load

90% reliability

50% reliability
Distributed vs Centralized

- Example: Poisson arrival, $r = \frac{2}{3}$

90% reliability

50% reliability
Wrap-up

1. **Strategy**: concurrent replication with canceling

2. **Model**: determine the response-time distribution
   - Insights into conditions affecting latency reduction
   - Allows to compare different set-ups

3. **Other Models**
   - Fork-join queue (Performance 2015)
   - Choice of $n \geq r$ servers (INFOCOM 2016)
   - $K$ out of $N$ tasks to finish (Erasure Coding)
THANK YOU