Java-based Simulation of Load Balancing Policies

Vaibhav Krishnakumar
June 2018

Supervisor: Giuliano Casale
Second Marker: Alex Carver

Distributed Systems

Scalability + Availability

Response Time / Latency

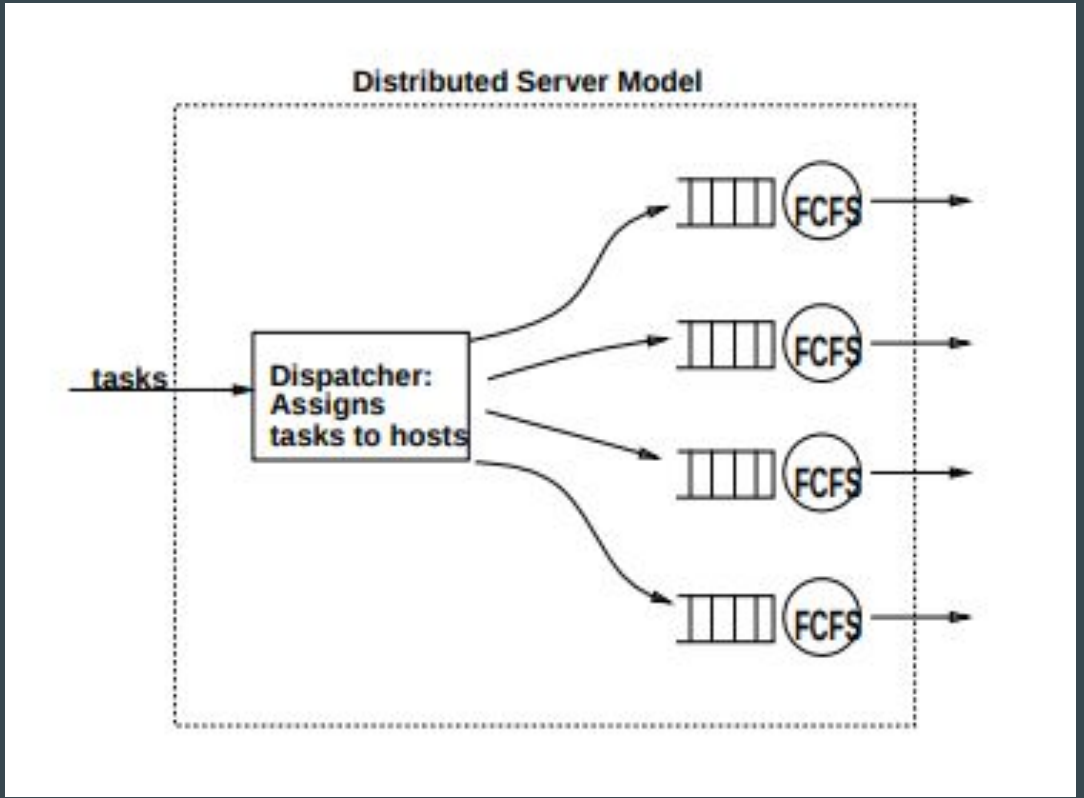
Optimal Weights

Load Balancing

Model

‘Supermarket’ Model

- Heterogeneous servers
- Multiple job classes
- Non preemptive tasks
- One-shot dispatch
- No redundant tasks
- FCFS
- ∞ capacity



Java Modelling Tools (JMT)

Politecnico di Milano, Imperial College

Capacity Planning, Queueing Network Simulations

Changes to codebase

- UI
- XML Model
- Routing Logic

50,000 users; 150,000 lines of code

Imperial College London

Load Balancing Policies

Random

Round Robin

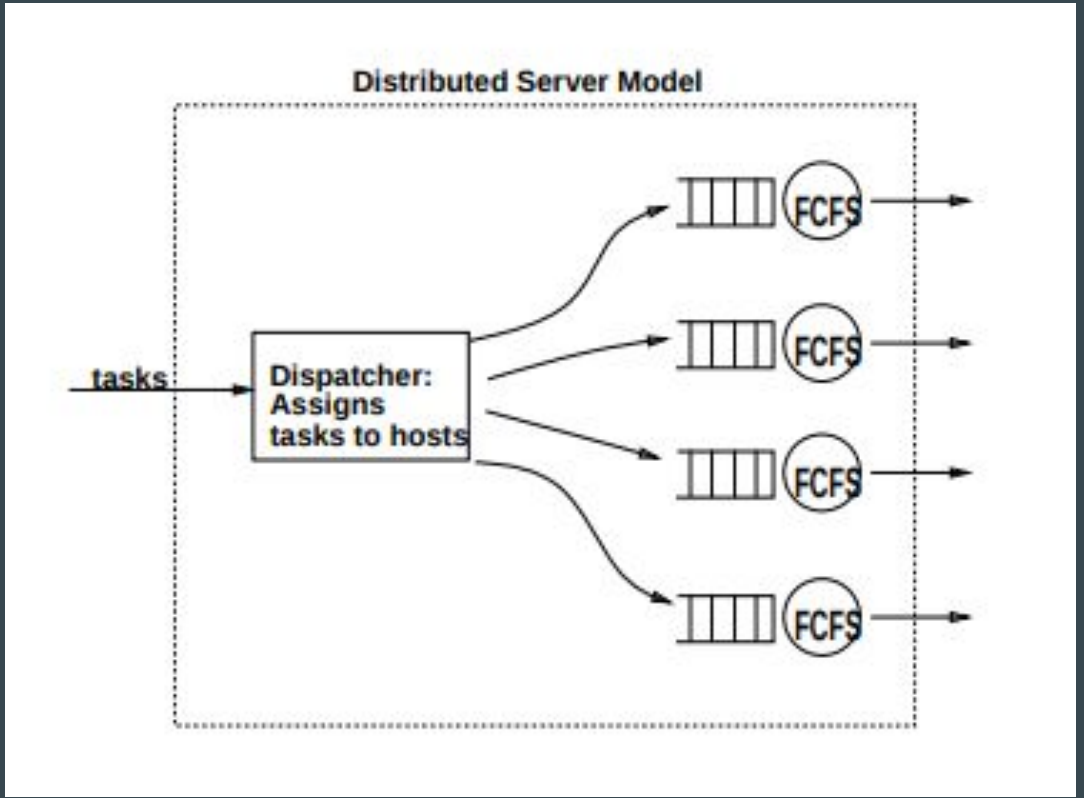
- Weighted Round Robin

Shortest Queue (JSQ)

- Power of k

Least Work Left

Size-Interval Task Assignment



Power of k

Performance optimisation to JSQ

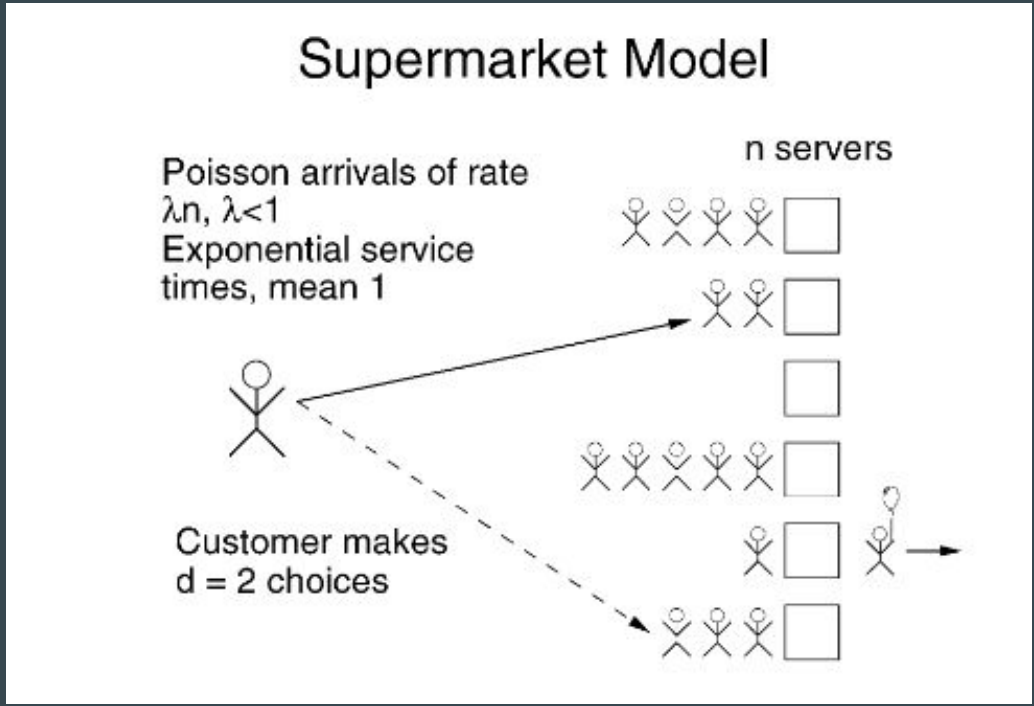
Query k < n servers on
each assignment

Send job to shortest observed queue

Trade-off optimality for performance

k = 1: Random

k = n: JSQ



Power of k with Memory

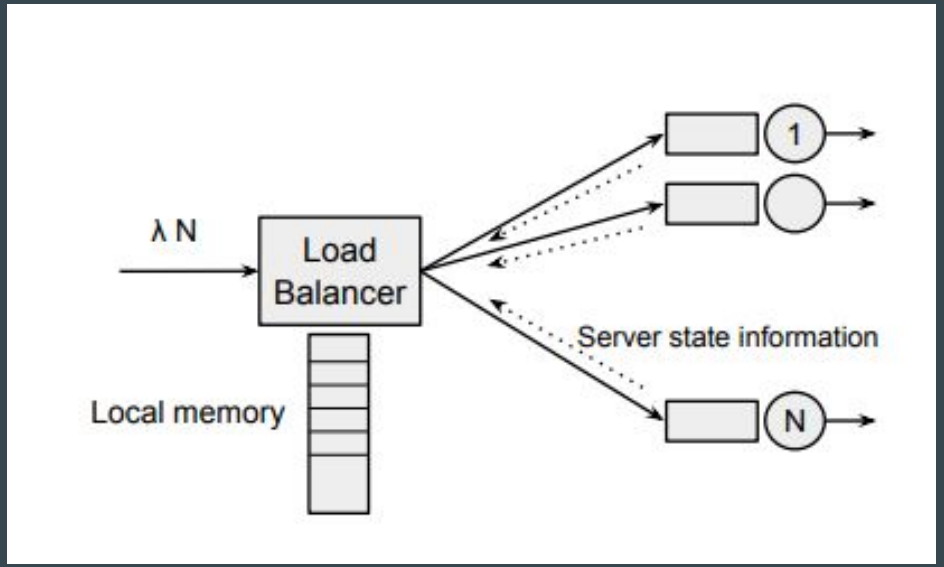
Add local memory to Power of k

Query k < n servers on each assignment as before

Store the results in local ‘memory’

Send task to best node in memory

It is possible to send the job to a node not queried in that round



Markov Chain Verification

Analytically solve a small, closed model with Power of k algorithms

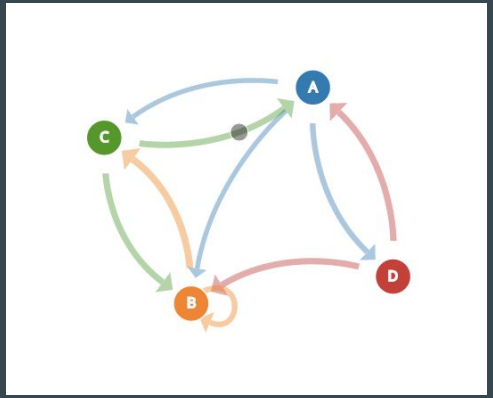
N = 4 for Power of k, N = 2 for ‘with Memory’

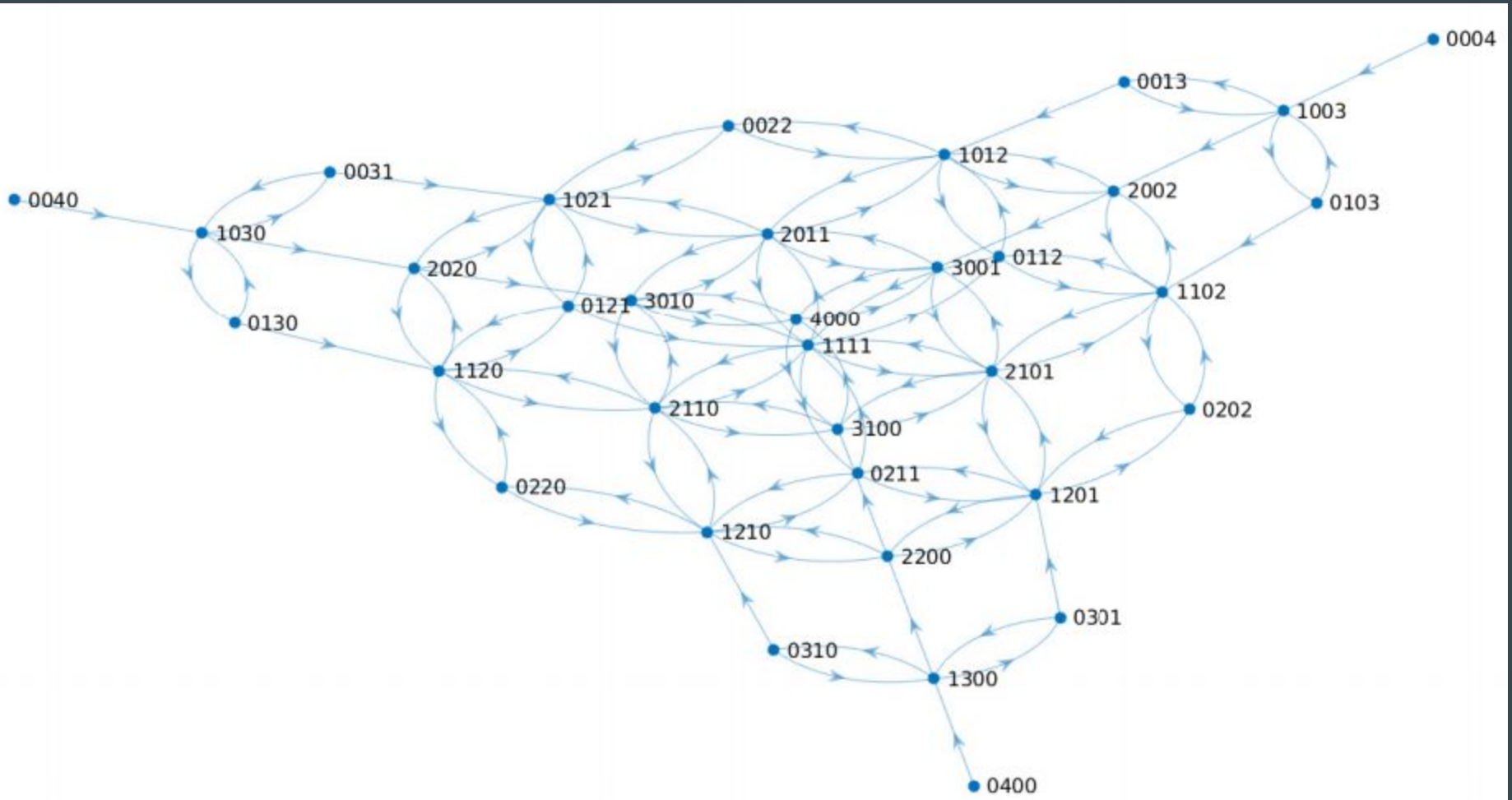
Solve flow-balance equation and normalise to obtain steady-state distribution

Calculate average queue length, utilisation and response time

Run simulation on JMT and compare

Results match closely, relative error under 1%




0004
0013
1003
0022
1012
0031
0040
1021
2002
0103
1030
2011
0112
2020
3001
1102
0121
3010
0130
4000
1111
1120
2101
2110
0202
3100
0211
0220
1201
1210
2200
0301
0310
1300
0400

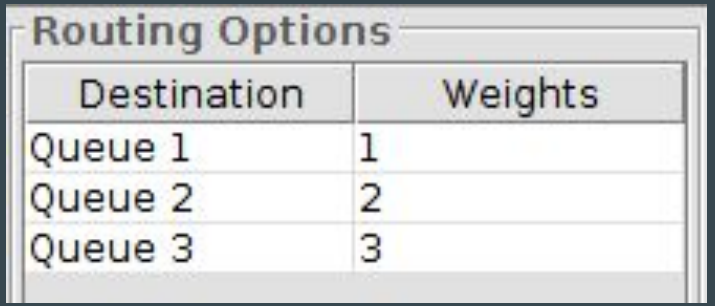
Weighted Round Robin

Cycle through servers, assigning jobs in proportion to weights

Weights are integers representing the number of jobs received in one full execution

Useful to exploit the heterogeneity of machines

Computing optimal weights is difficult in practise

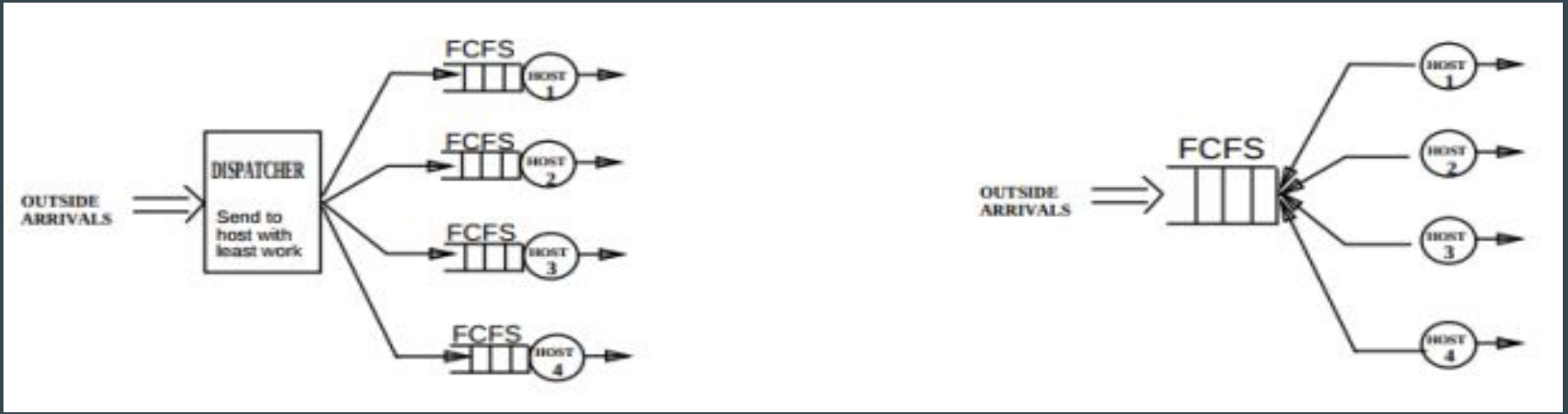


Least Work Left

Calculate the `WORK_LEFT` metric for each server

- Sum of service times of jobs waiting in queue
- Residual service time of job in service

Central Queue



SITA

Partitioning the job size distribution

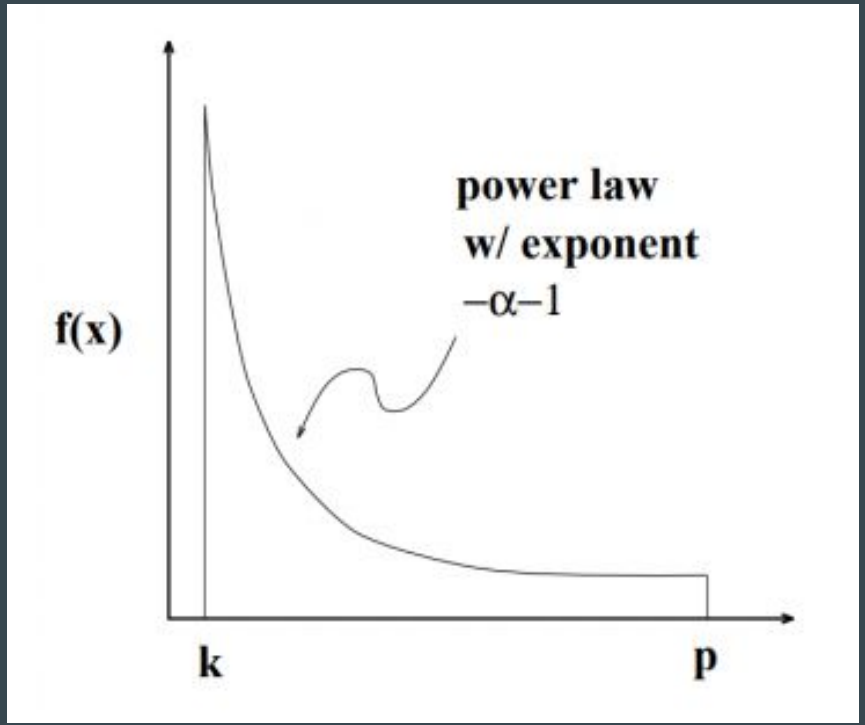
Allocate a server per partition

Cut-offs vary by objective

- SITA-OPT
- SITA-E

Bounded Pareto Distribution

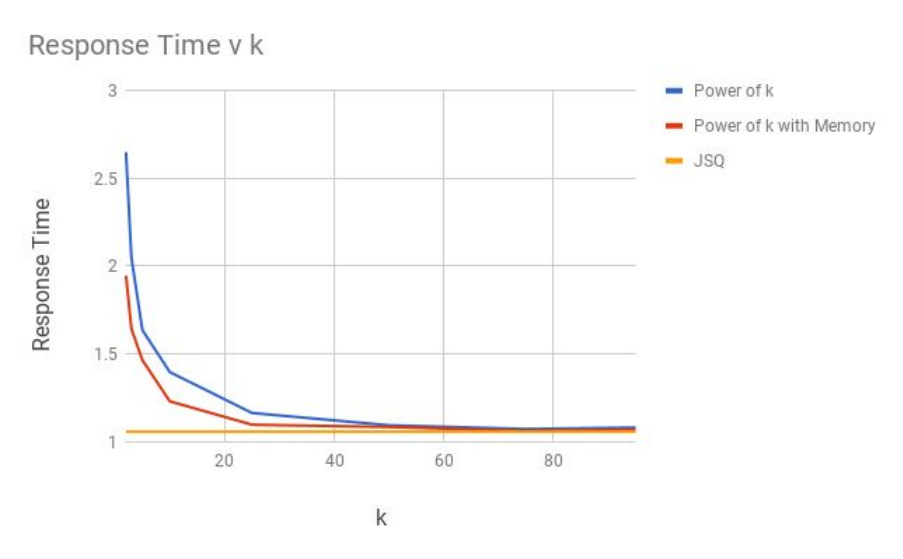
- File size on the internet

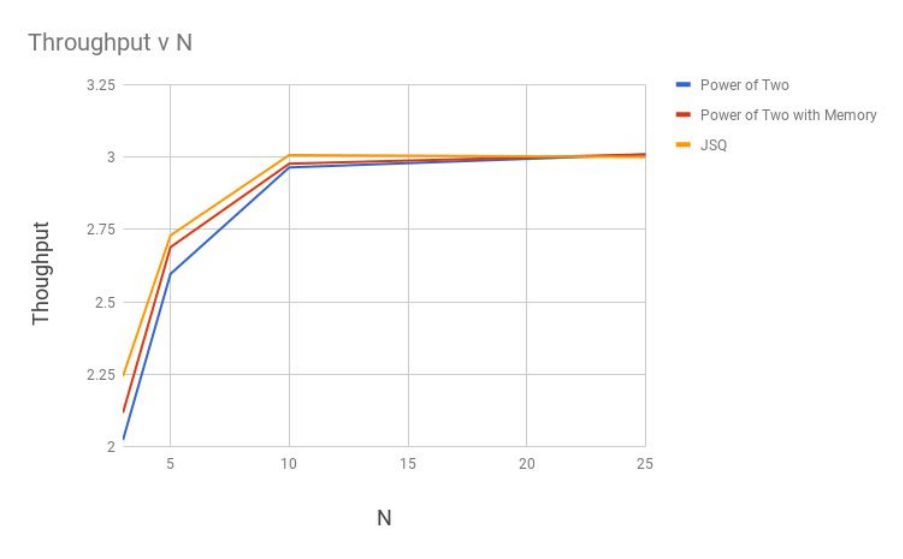


The impact of adding memory

JSQ is the bounding case of Power of k strategies

The performance converges as number of choices (k) or number of jobs (N) increase



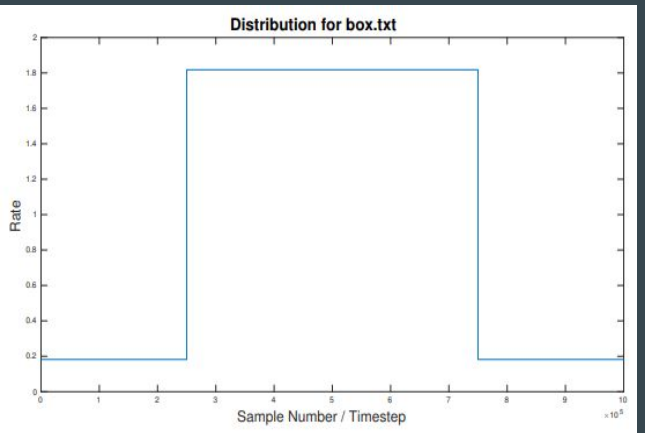


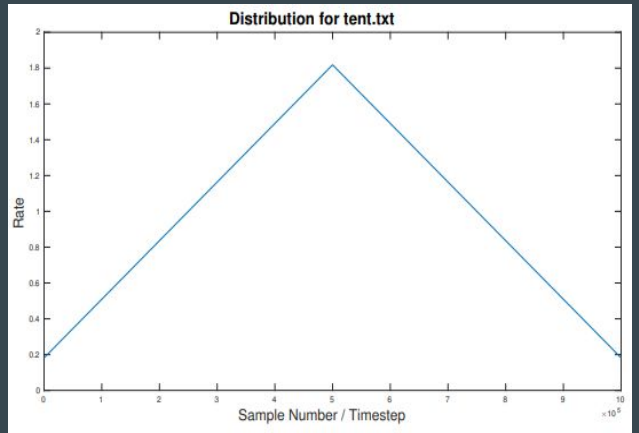
Resilience to Inter-arrival Times

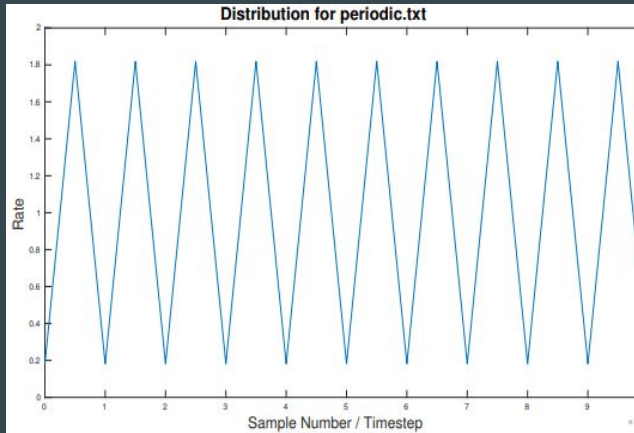
Vary inter-arrival rates based on distributions below

LWL and Power of k with Memory reacted best to the changes

Other algorithms perform similarly, further research required to differentiate







Criteria

Response Time Minimisation

Throughput Maximisation

Utilisation Maximisation

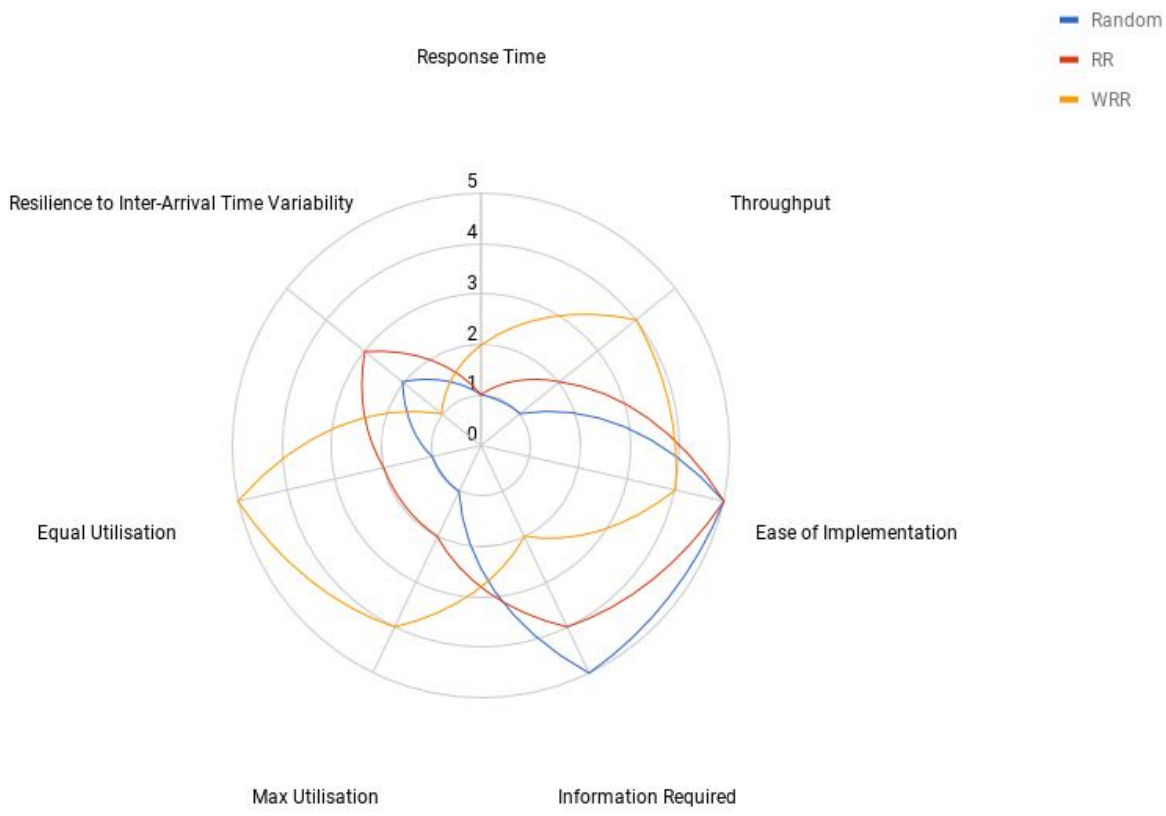
Utilisation Equalisation

Resilience to Inter-arrival Time Variability

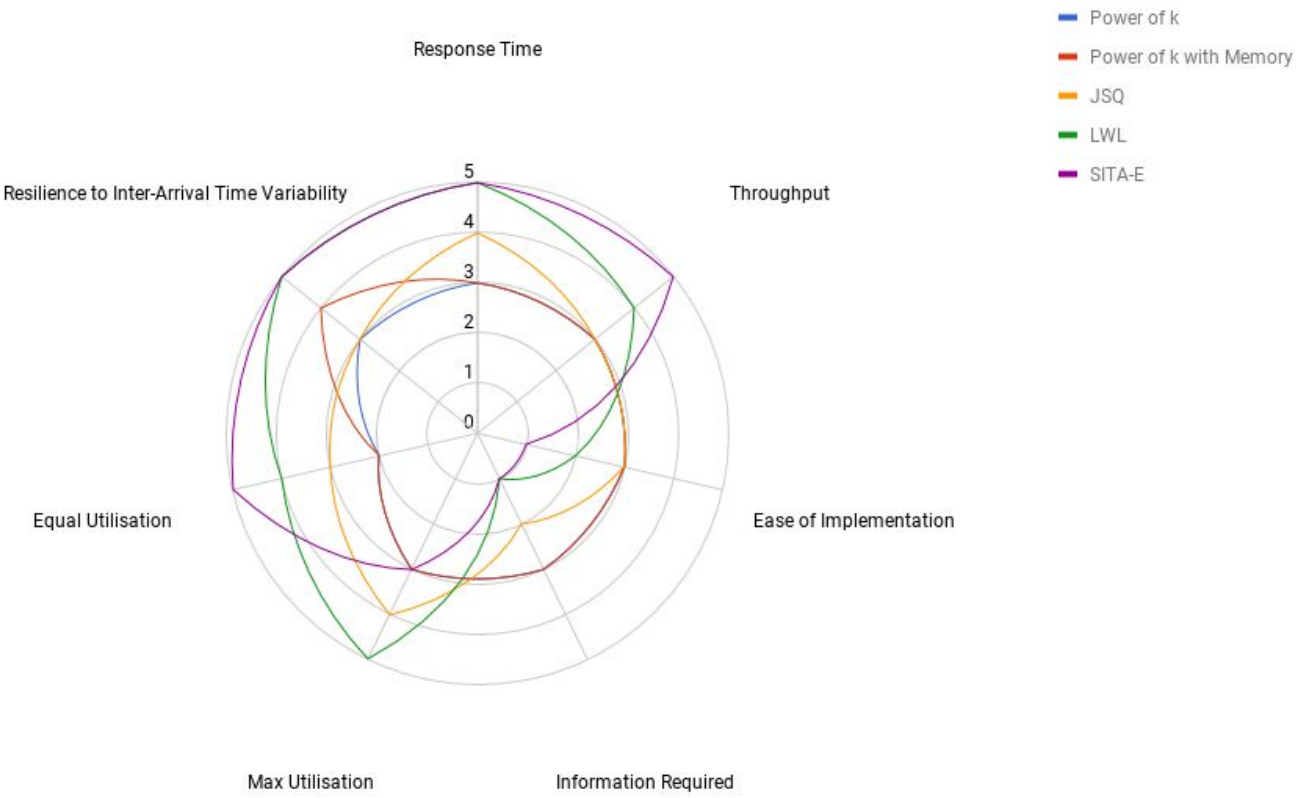
Information Required

Ease of Implementation

Static Policies



Dynamic Policies



Evaluation

Power of k - Markov Chain modelling

Power of k with Memory

- Demonstrate results from original paper
- Prof. Anselmi ran further tests, performs as expected

Least Work Left

- Further testing needed
- Verification is not ‘convincing’

Future Work

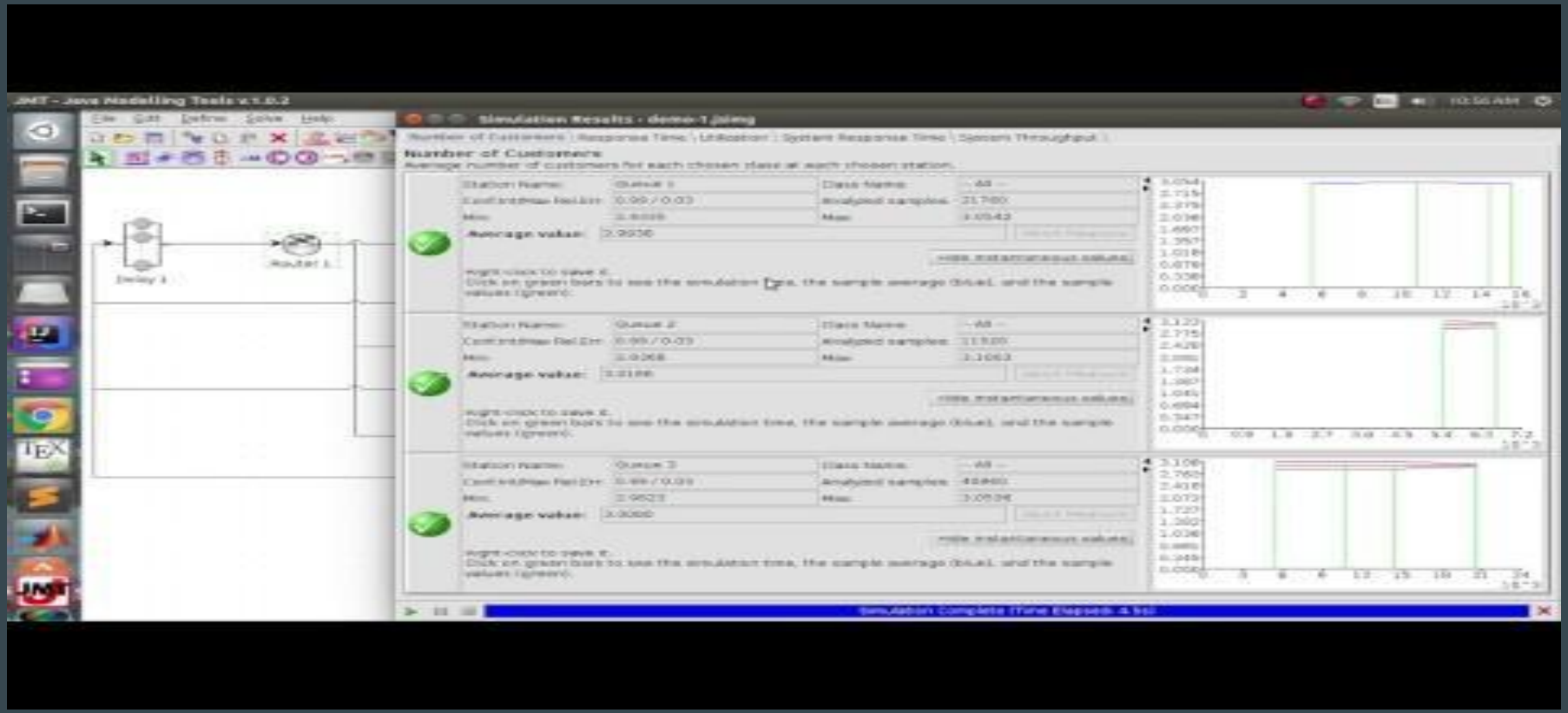
Automate Markov Chain generation

- Simple program to create transition matrices
- Larger, more complex models can be analysed with ease

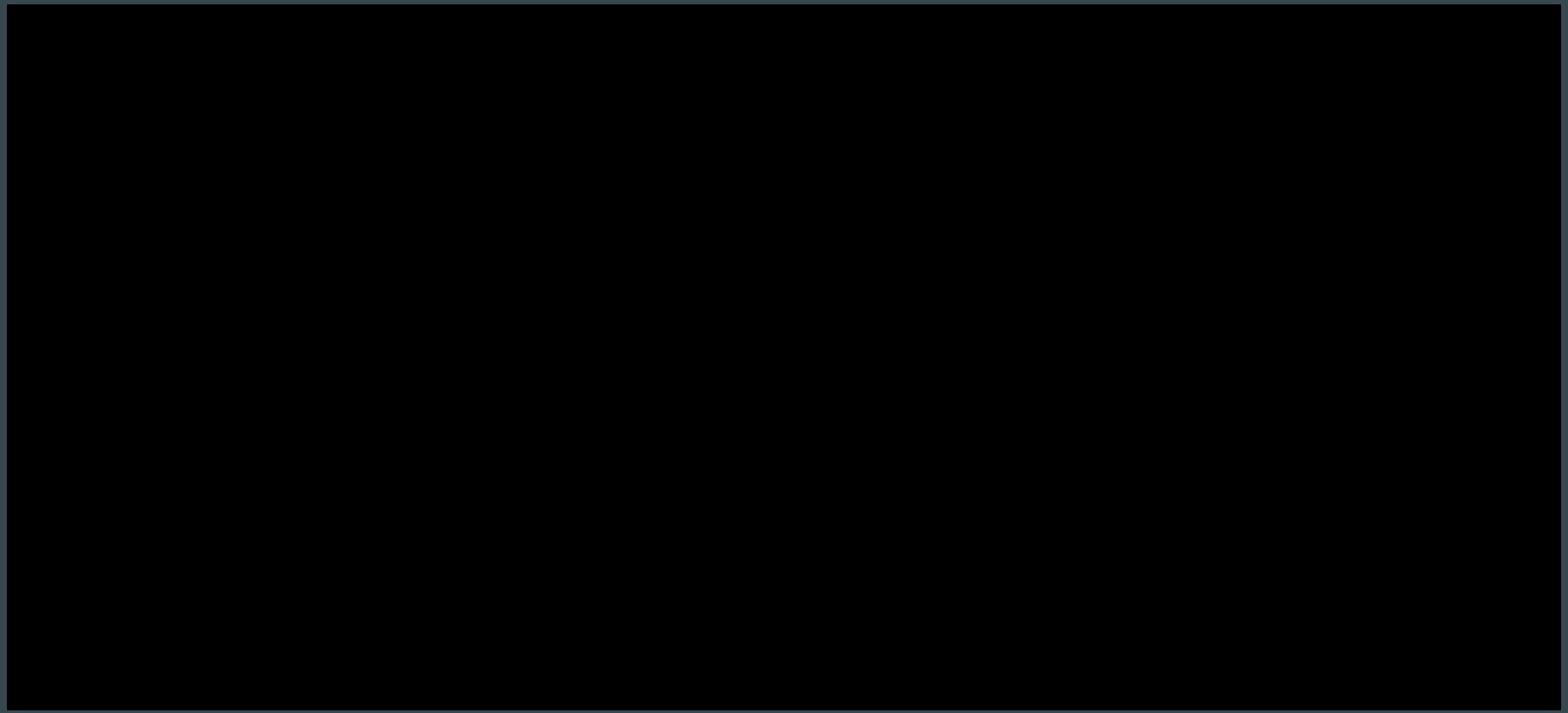
SITA

- Integrate into JMT properly
- Introduce concept of job size
- Collaboration of servers

Demo - Power of k, Power of k with Memory



Demo - Weighted Round Robin, Least Work Left



Demo - SITA