Aggregate Architecture Simulation in Event-Sourcing Applications using Layered Queuing Networks

Gururaj Maddodi, Slinger Jansen, and Michiel Overeem

Utrecht University
VU VRIJE UNIVERSITEIT AMSTERDAM

AMUSE Project

Introduction

- Unlike traditional software frameworks, Event Sourcing framework is dynamic in Performance Engineering sense in that each request can have different resource demands

- The requests in Event Sourcing systems have to replay all the change to the state (called events) to execute, which causes requests to have different resource demands

- In this paper an approach to model this behavior along with two possible architecture is investigated

Research Method

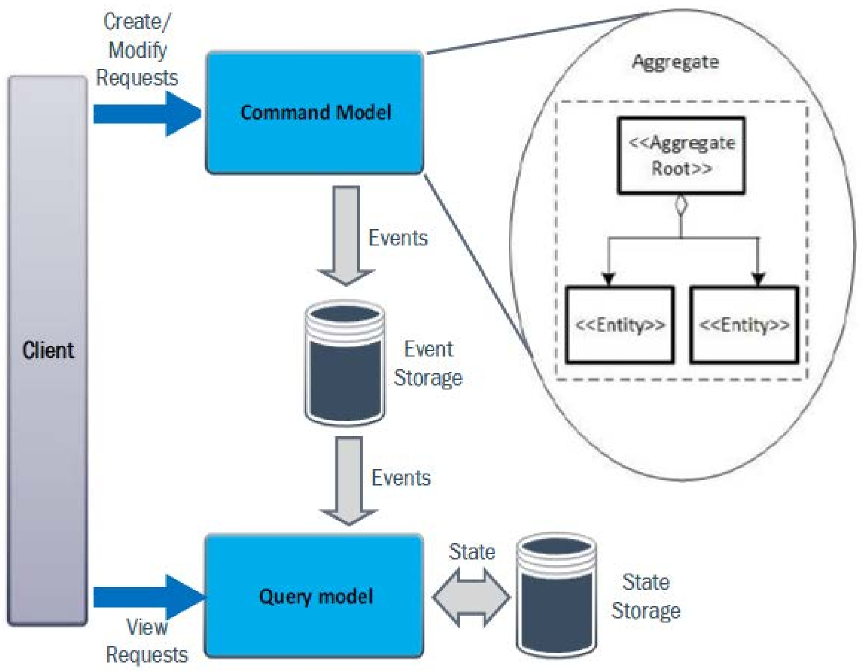
- Software Performance Engineering (SPE) Methodology, Describe Software Execution Model (UML sequence diagram, Execution graph ...)

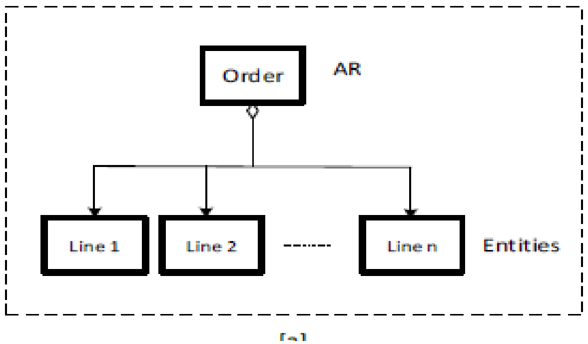
  to

  Derive System Execution Model (Queuing network, Process algebra...)

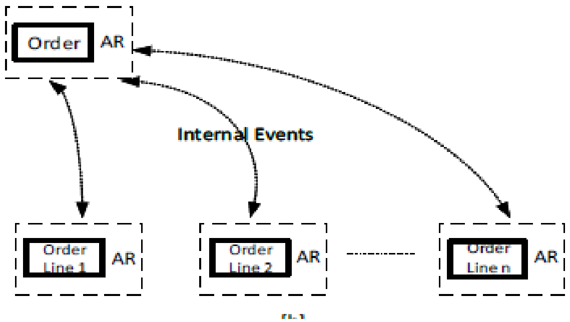
- Use measurements to fill the individual components of the model which could then be used to predict performance metrics of the overall system

Event Sourcing (and possible architectures)





Single Aggregate



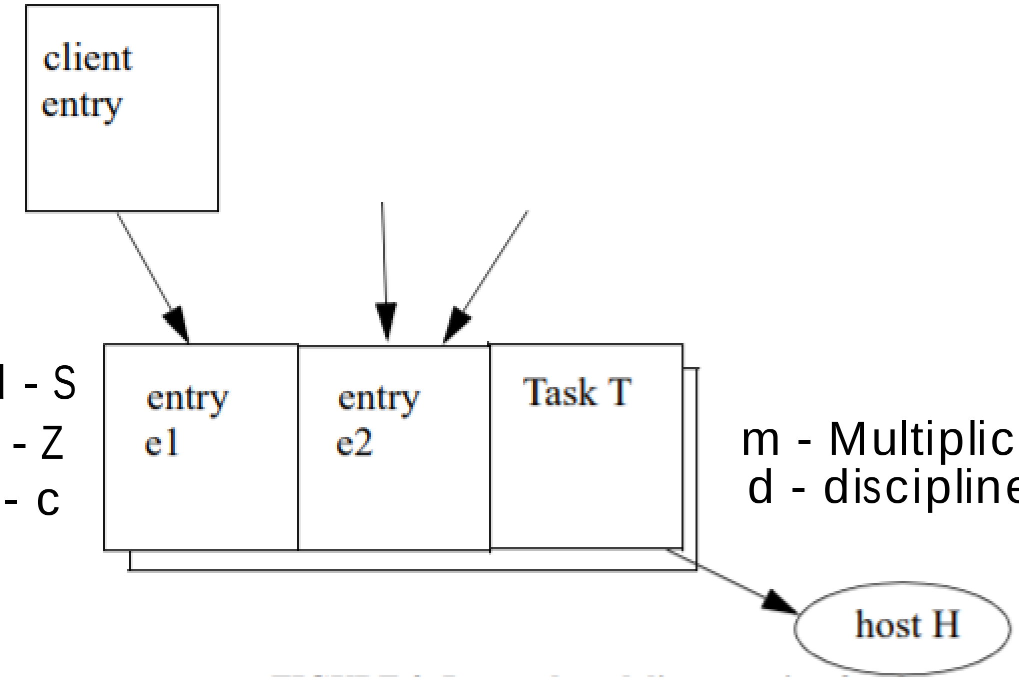
Multiple Aggregate

Layered Queuing Networks

- Queuing theory is the mathematical study of the congestion and delays of waiting for requests
- QN models do not account for software contention, i.e. stations are not considered as both client and servers
- Layered Queuing Networks simulate systems with communication between the stations

Layered Queuing Networks

Service Demand - S
Delay - Z
Avg. no. of calls - c



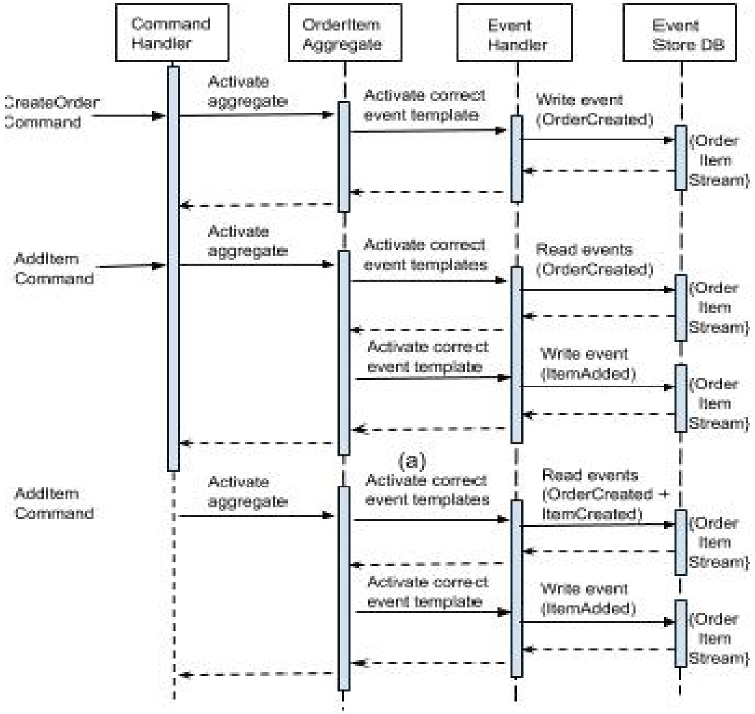
m - Multiplicity
d - discipline

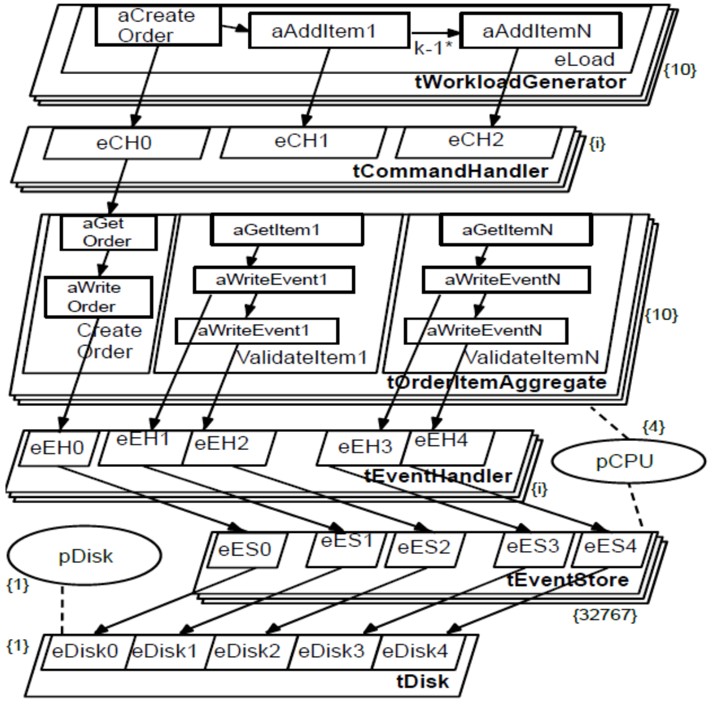
m - Multiplicity
d - discipline

Modeling Components

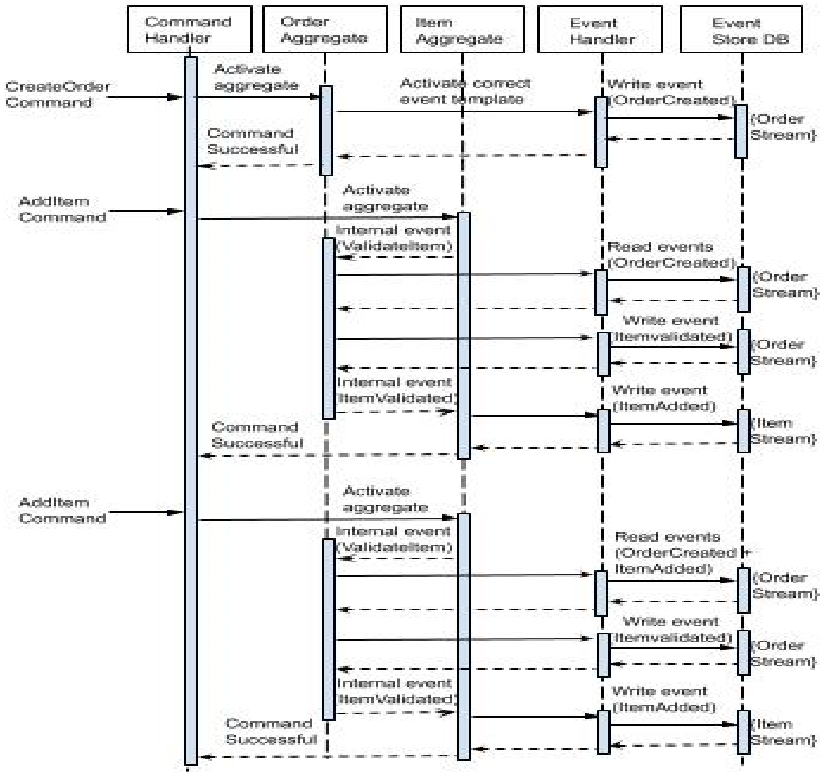
- CPU and disk as processors representing actual hardware resources
- Aggregates as task/s with entries containing resource demands only on the CPU processor as it only works on the events available in memory
- Event and Command Handlers as tasks containing only delays and no resource demands as they just redirect requests
- Event store or database that has resource demand on CPU processor and also connects to disk processor through disk task
- Disk task to simulate disk handlers for different requests

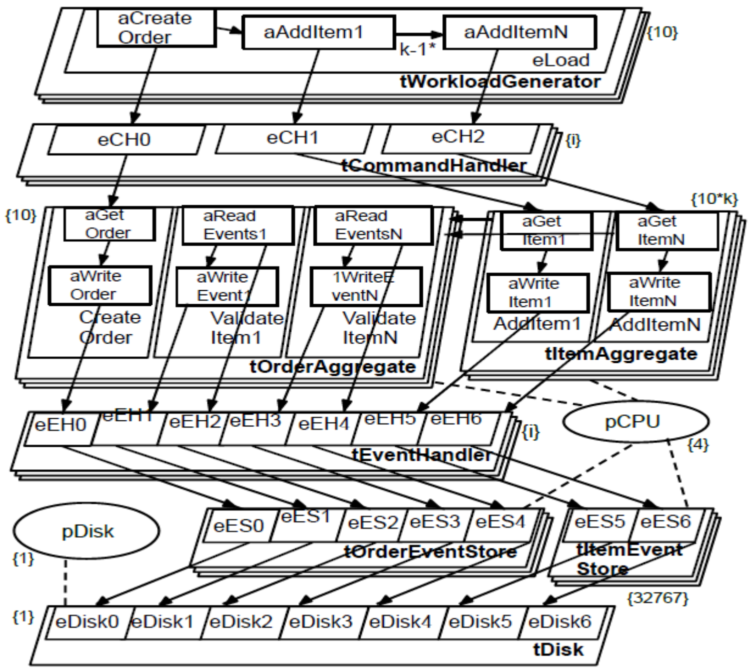
System Execution Model (Single Aggregeate)





System Execution Model (Multiple Aggreegate)





Measuring Service Demands

- Service demand is the time that a specific request class spends at the resource that is servicing it

- Queuing theory specifies rules to calculate it
  - Little's Law
  - Utilization Law (specific case of Little's Law)
  - Forced Flow Law

Utilization Law

- Utilization law:

$$D_{c,i} = U_{c,i} / X_c$$

$U_{c,i}$ – utilization at the station/s at a resource

X_c – Throughput of whole system

Forced Flow Law

- Forced Flow Law

$$V_{c,i} = X_{c,i} / X_c$$

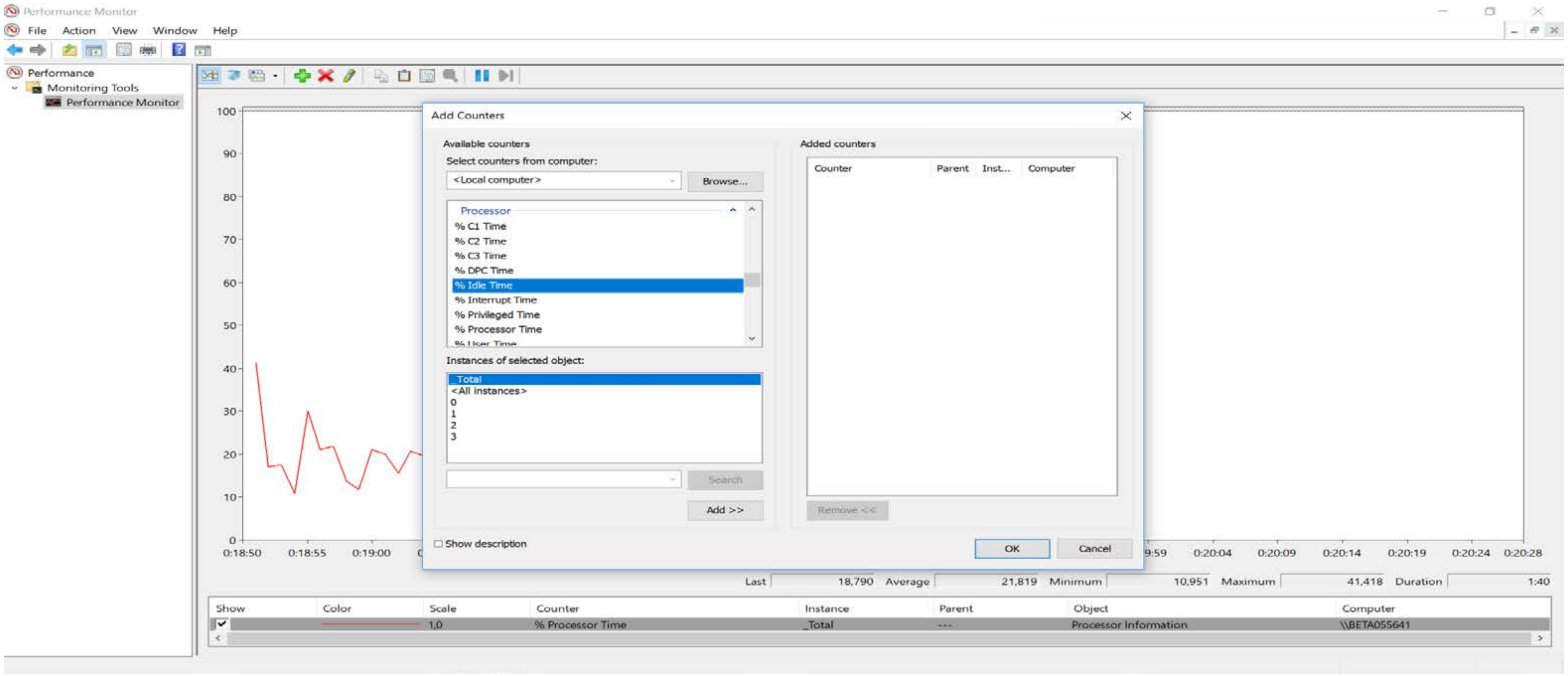
$V_{c,i}$ – no. of visits at station i

$X_{c,i}$ – Throughput at station i

$$D_{c,i} = V_{c,i} * S_{c,i}$$

$S_{c,i}$ – service time per visit at station i

Measurement Tools



Performance Monitor

- Counters that can be used:
  - % CPU idle
  - % disk time
  - Disk tps

  - % CPU SQL Server instance
  - Transactions/sec
  - Write transactions/sec

  - Overall throughput
  - Request processing time

Applying Queuing Theory Laws

- Calculate service demand for CPU resource

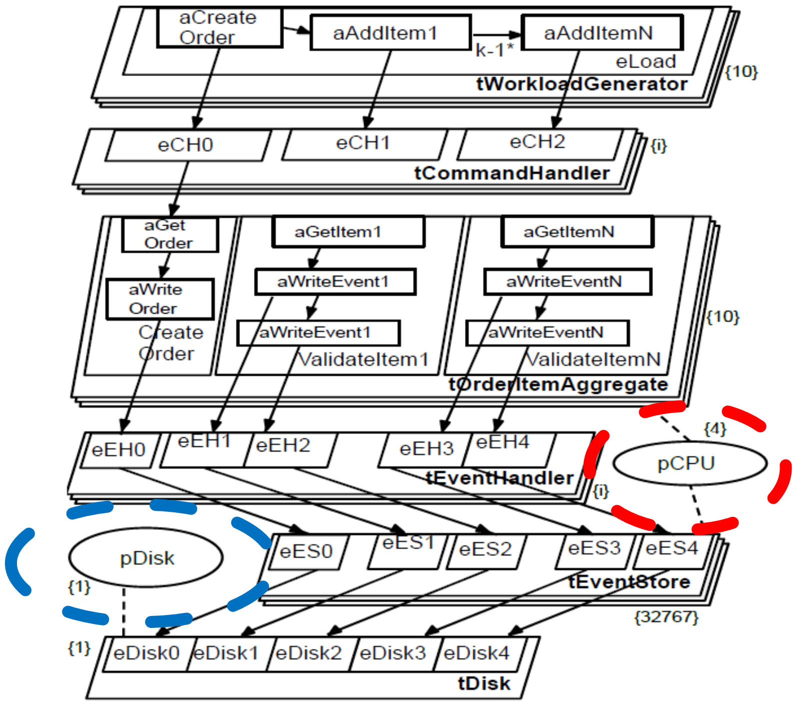
  CPU Utilization / Overall throughput

- Service demand for disk resource

  Disk Utilization / Overall throughput

  Or

  Disk tps / Overall throughput * service time / request



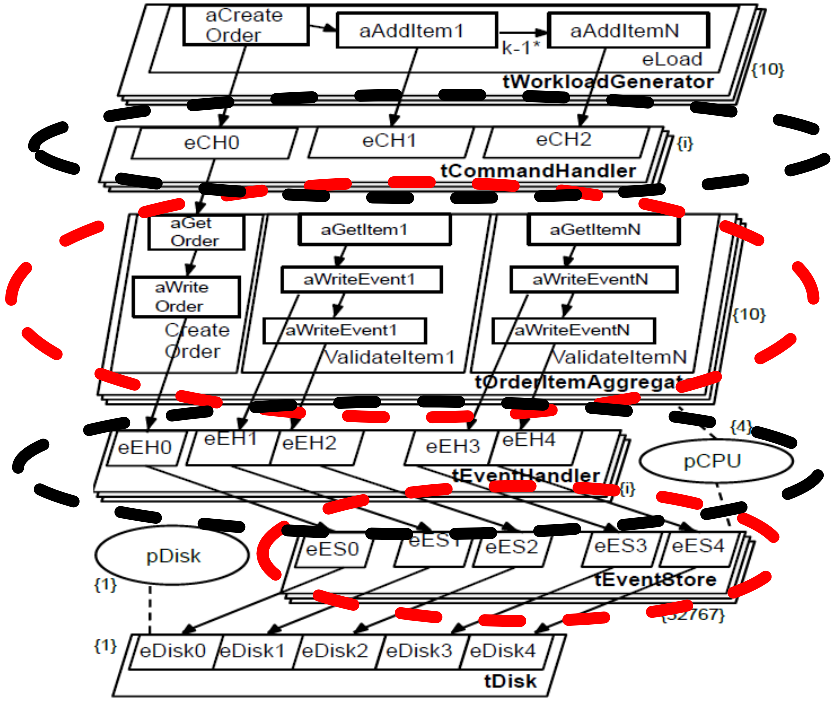
Applying Queuing Theory Laws

- Service demand for CPU resource only DB

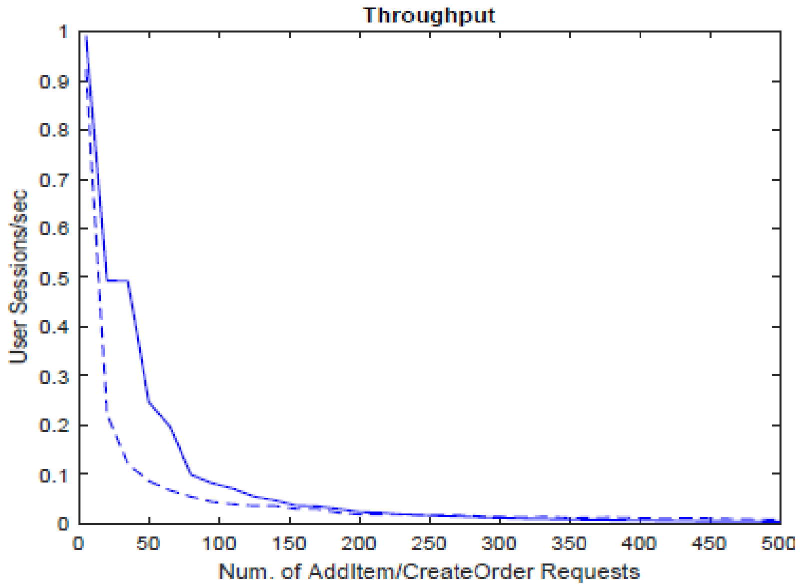
  CPU Utilization of DB / Overall throughput

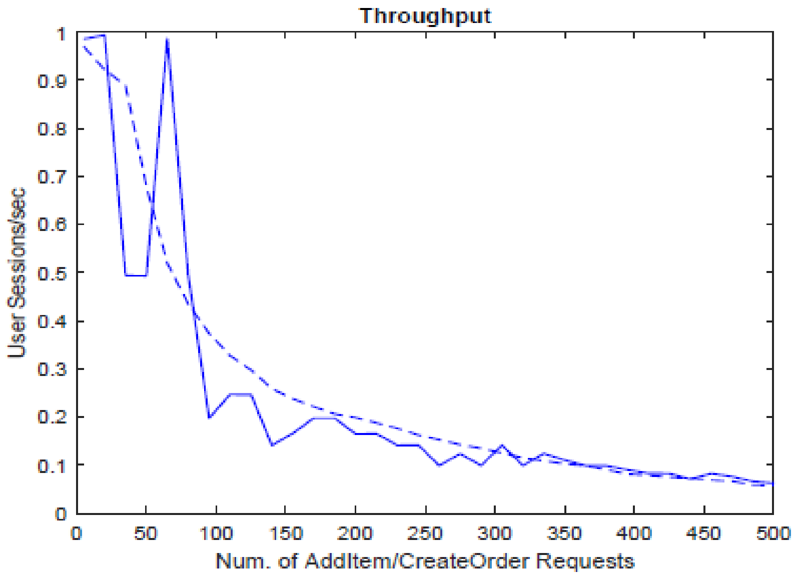
- Application Service Demand = total execution time – DB service demand

- Remaining from the total execution time is delay

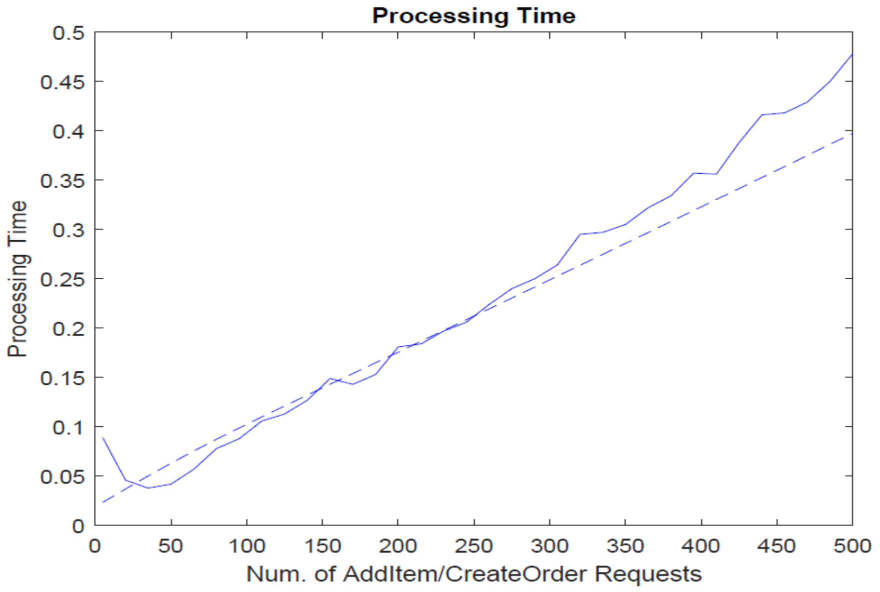


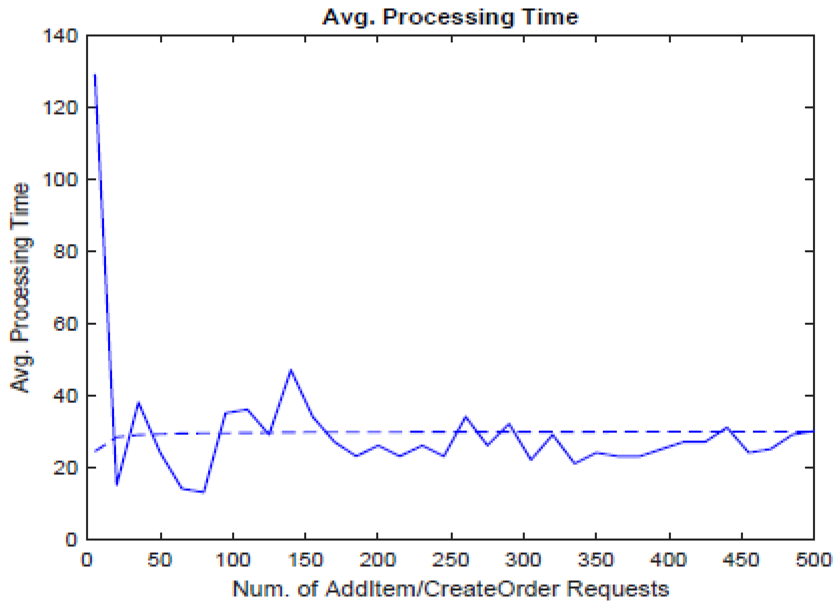
Measurement vs. Prediction





Measurement vs. Prediction





Conclusion and Future Work

- Event Sourcing systems can be modeled by spreading out the resource demand of the request with largest number of events evenly across entire life cycle of aggregate
- The two possible architectures can be simulated by separating out tasks such as aggregate and event store so that separate queues handle the requests
- Future work envisioned is using the modeling technique described here and applying it on larger systems

Thanks for Listening