Performance Analysis of a Multi-Tenant In-memory Data Grid

Anwesha Das[Ψ], *Frank Mueller*[Ψ], *Xiaohui Gu*[Ψ], *Arun Iyengar*[φ]

[Ψ]North Carolina State University
[φ]IBM T. J. Watson Research Center

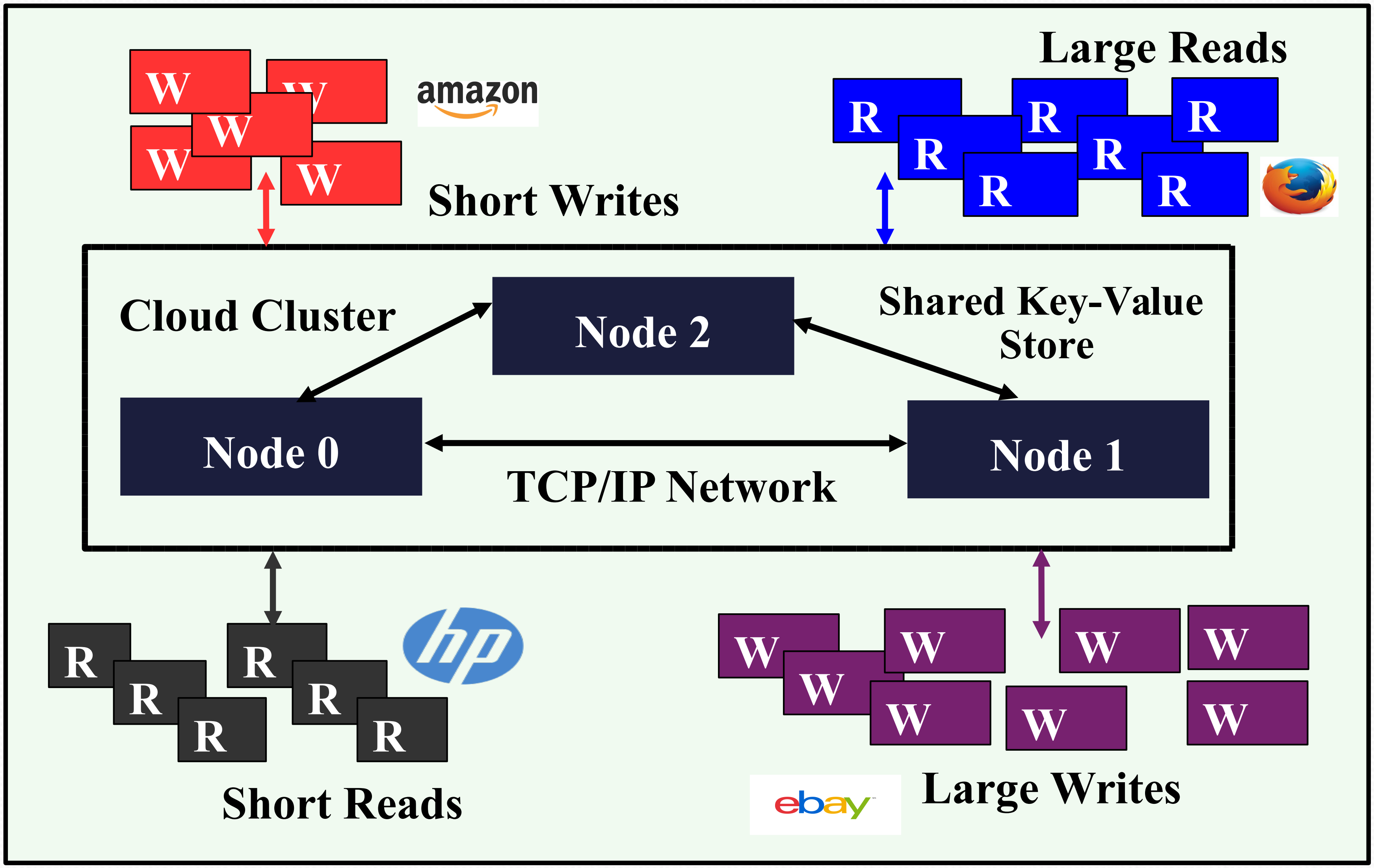
Key-Value Store Users

Key-Value Stores	**Open Source**
BigTable	*No*
Pnut	*No*
DynamoDB	*No*
MongoDB	*Yes*
Voldemort	*Yes*
HBase	*Yes*
HyperTable	*Yes*
ZBase	*Yes*
Cassandra	*Yes*
MemcacheD	*Yes*
Redis	*Yes*
Hazelcast	*Yes*
Comet	*Academic*
Silt	*Academic*

Wide Commercial and Academic Usage

Motivation

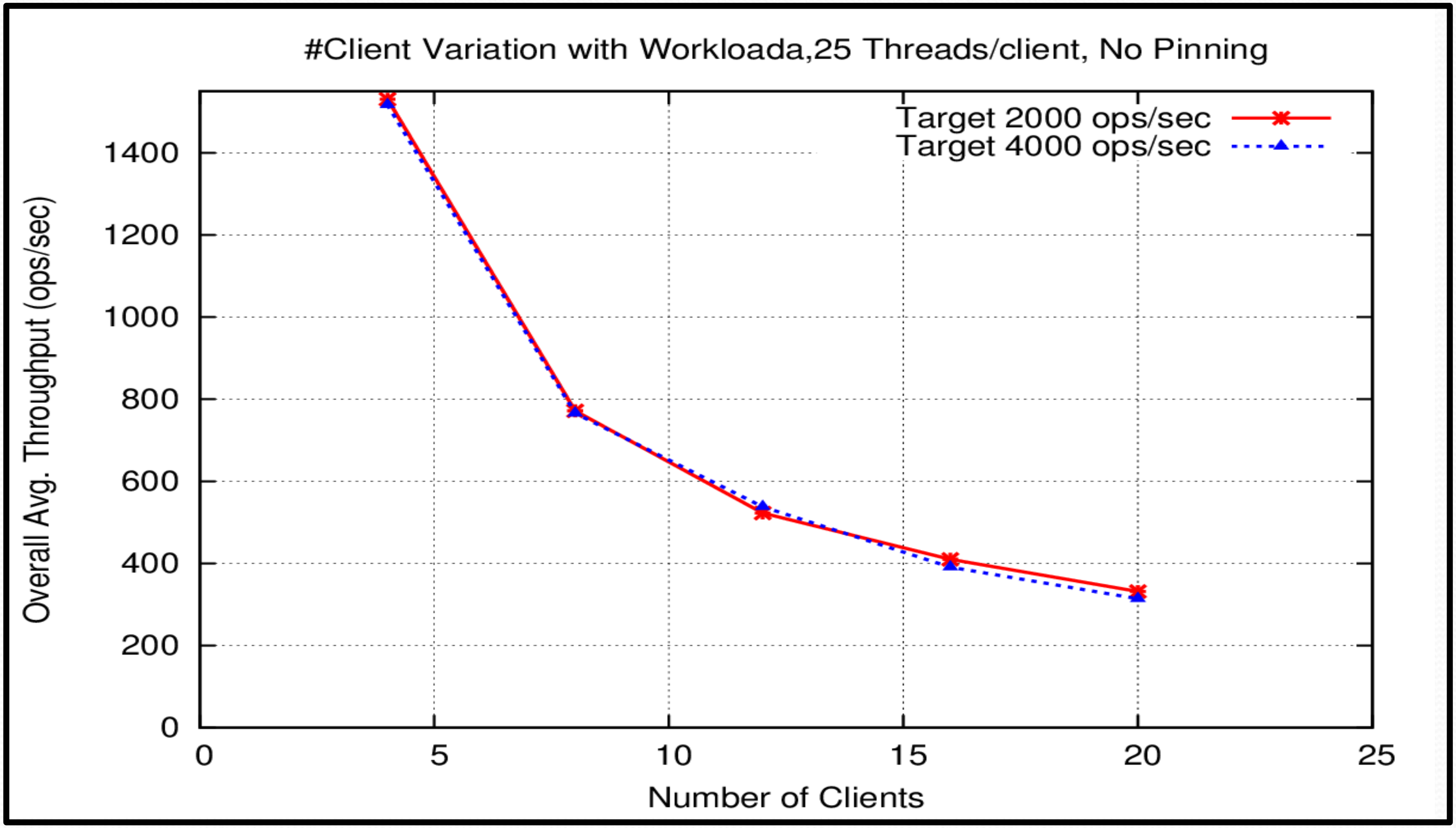
- Variable Unpredictable Workload Dynamics
- Problem ? Multi-Tenant Interference



Objectives

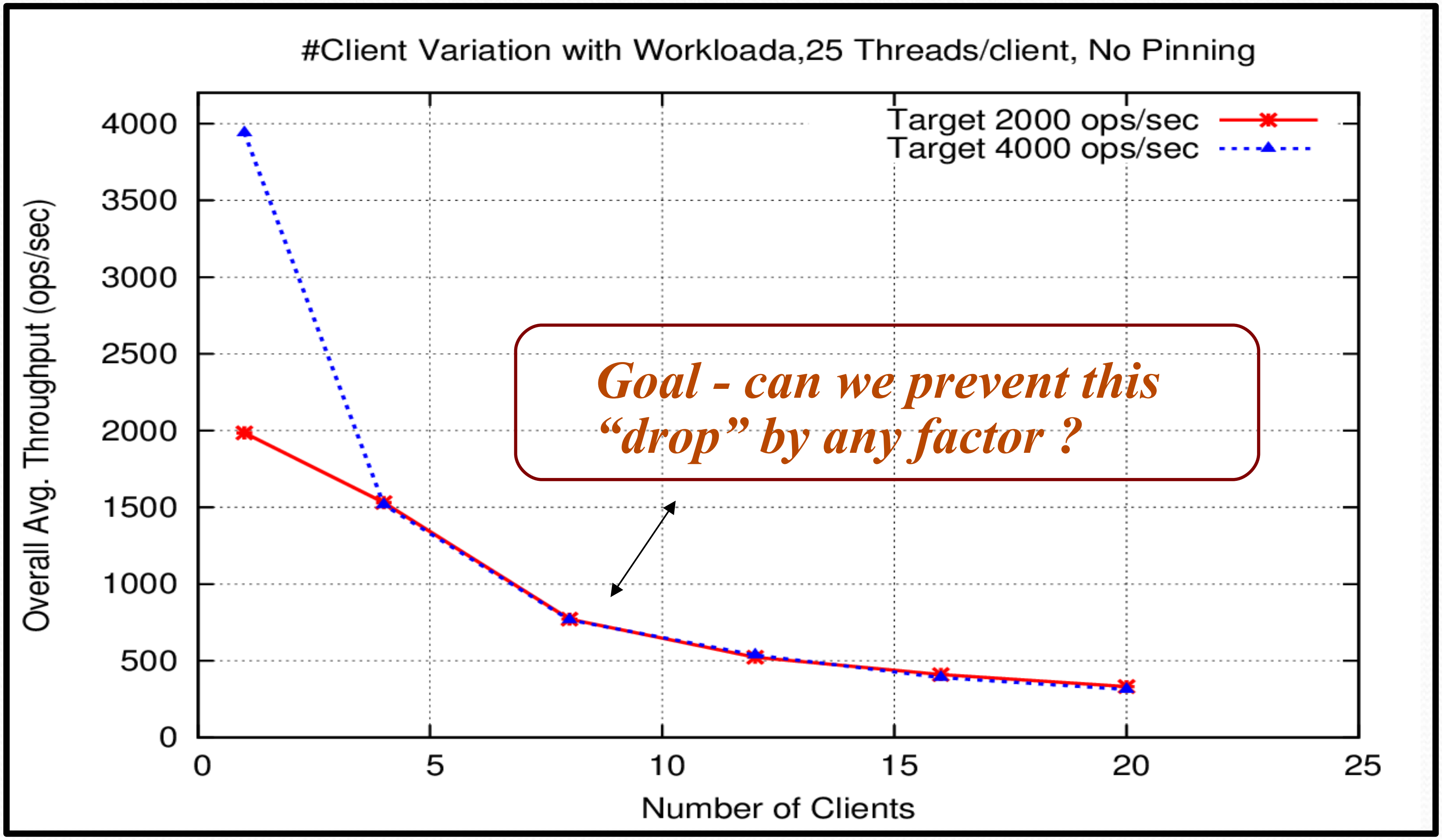
- ***Understand Hazelcast in the context of multi-tenancy.***
- ***How high is the performance degradation ?***
- ***Observe performance characteristics with varying number of clients, workload type, target throughput, thread count.***
- ***What causes degradation ? Investigate performance bottlenecks in Hazelcast to eliminate contention.***

Problem



- ✓ ***Decrease in throughput (ops/sec) with increase in number of clients***
- ✓ ***Presence of contention leading to performance degradation***

Problem



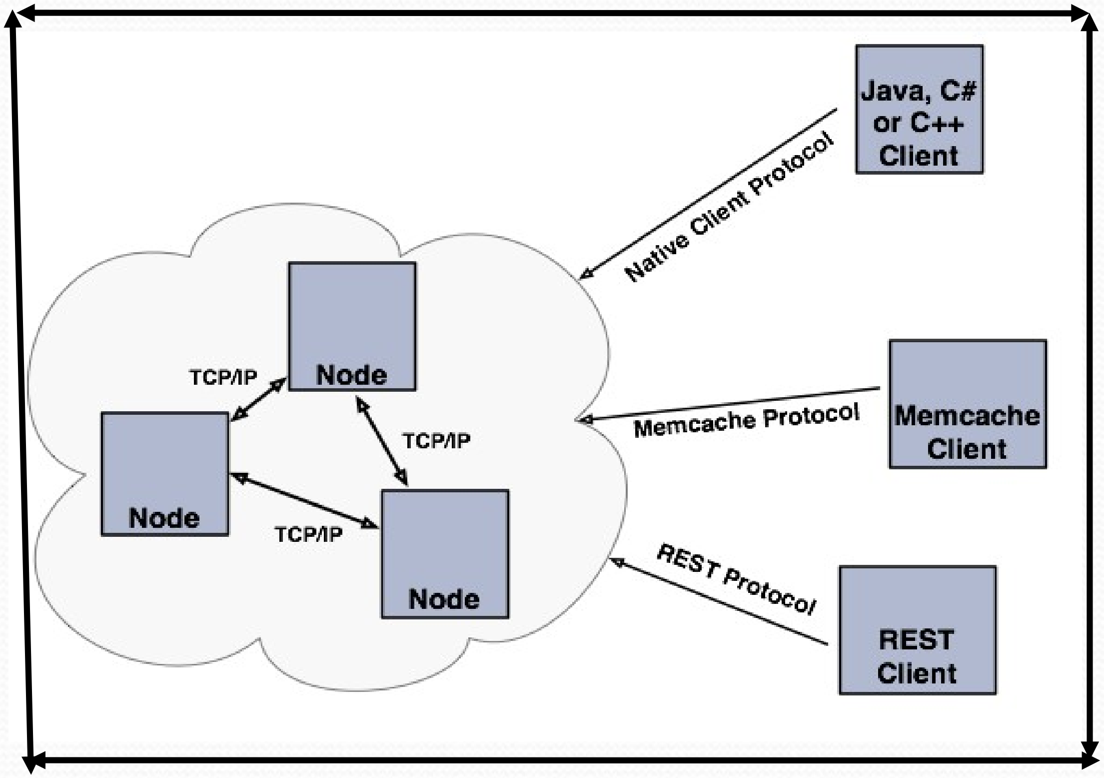
Problems ? *Resource Contention, Performance Degradation*

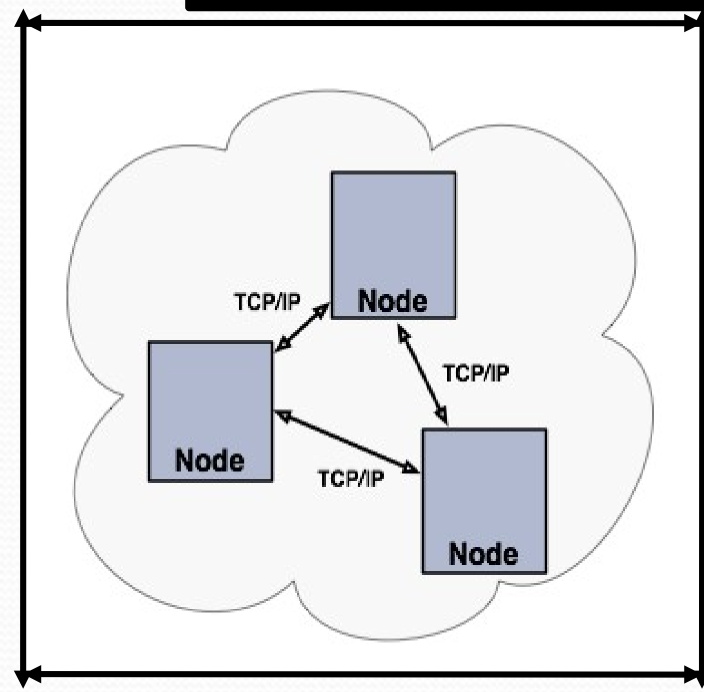
Aim *– Understand the source of contention, Find out performance bottlenecks in Hazelcast, How to alleviate contention ??*

Hazelcast Architecture

Client to Cluster

Node contains Data Primary & Replica





Peer to Peer

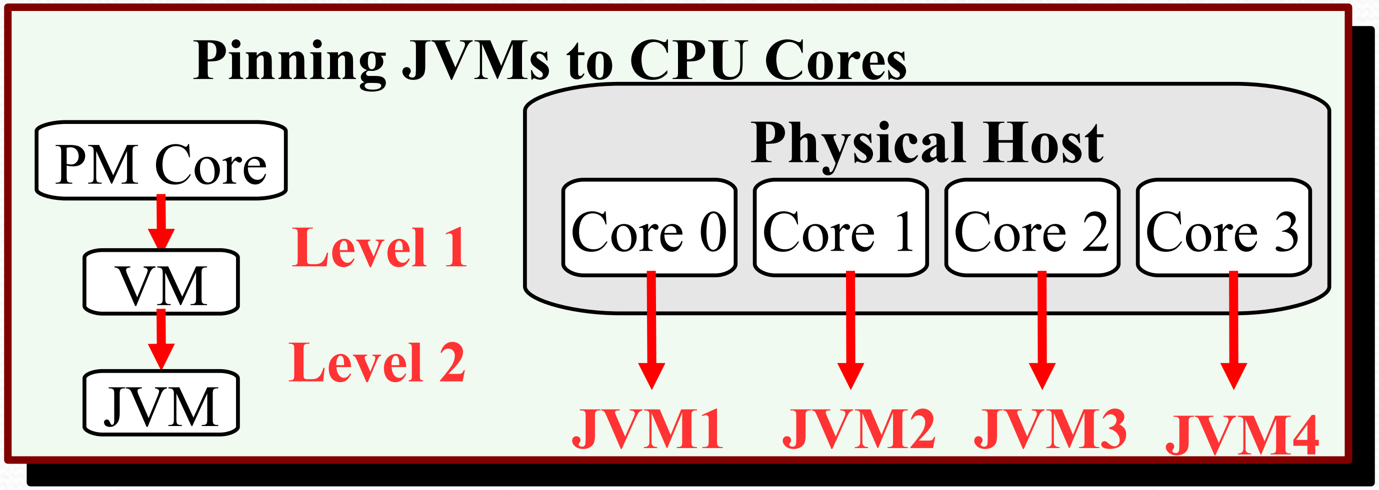
Hazelcast

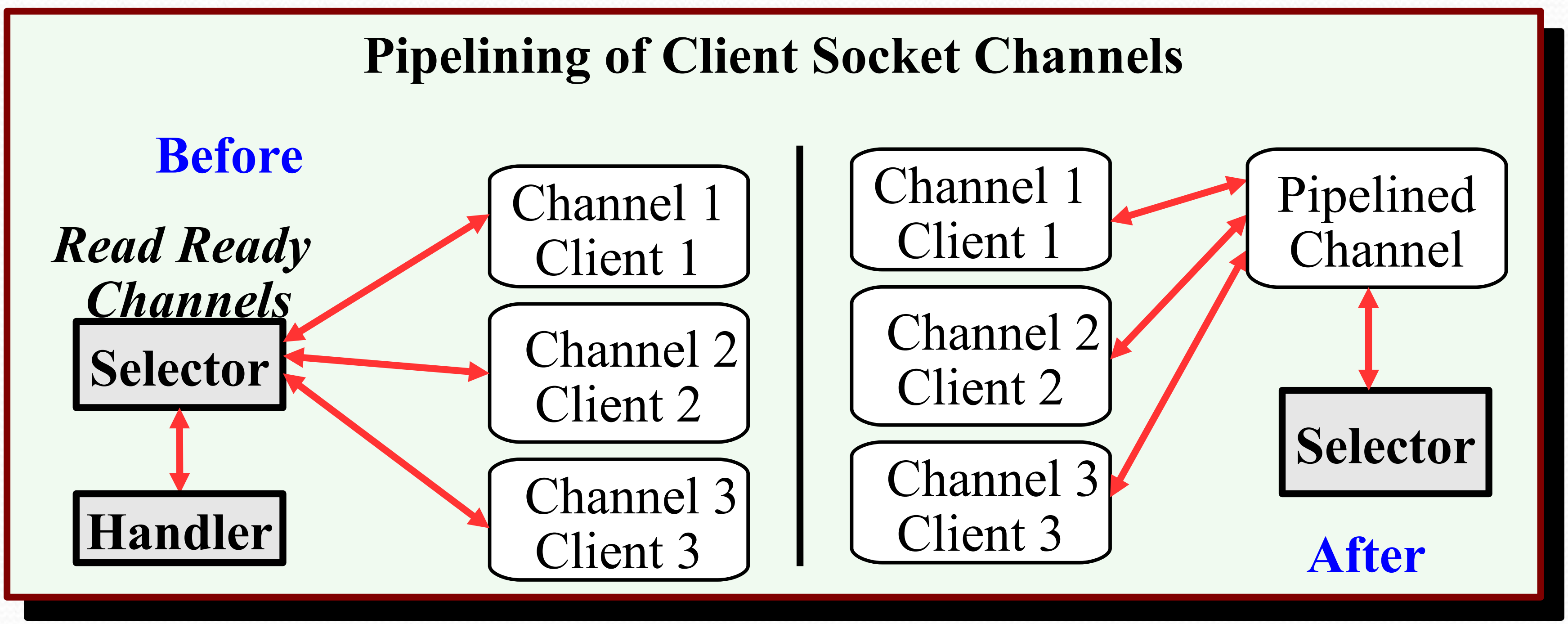
Client – No data
Default – 271 Partitions

- **Open-source In-memory Data Grid**
- **Decentralized Architecture**
- **TCP/IP based communication between nodes**

Solution Approach

- Pin JVM to CPU Cores
- Coalesce Socket Channels





Implementation

Software

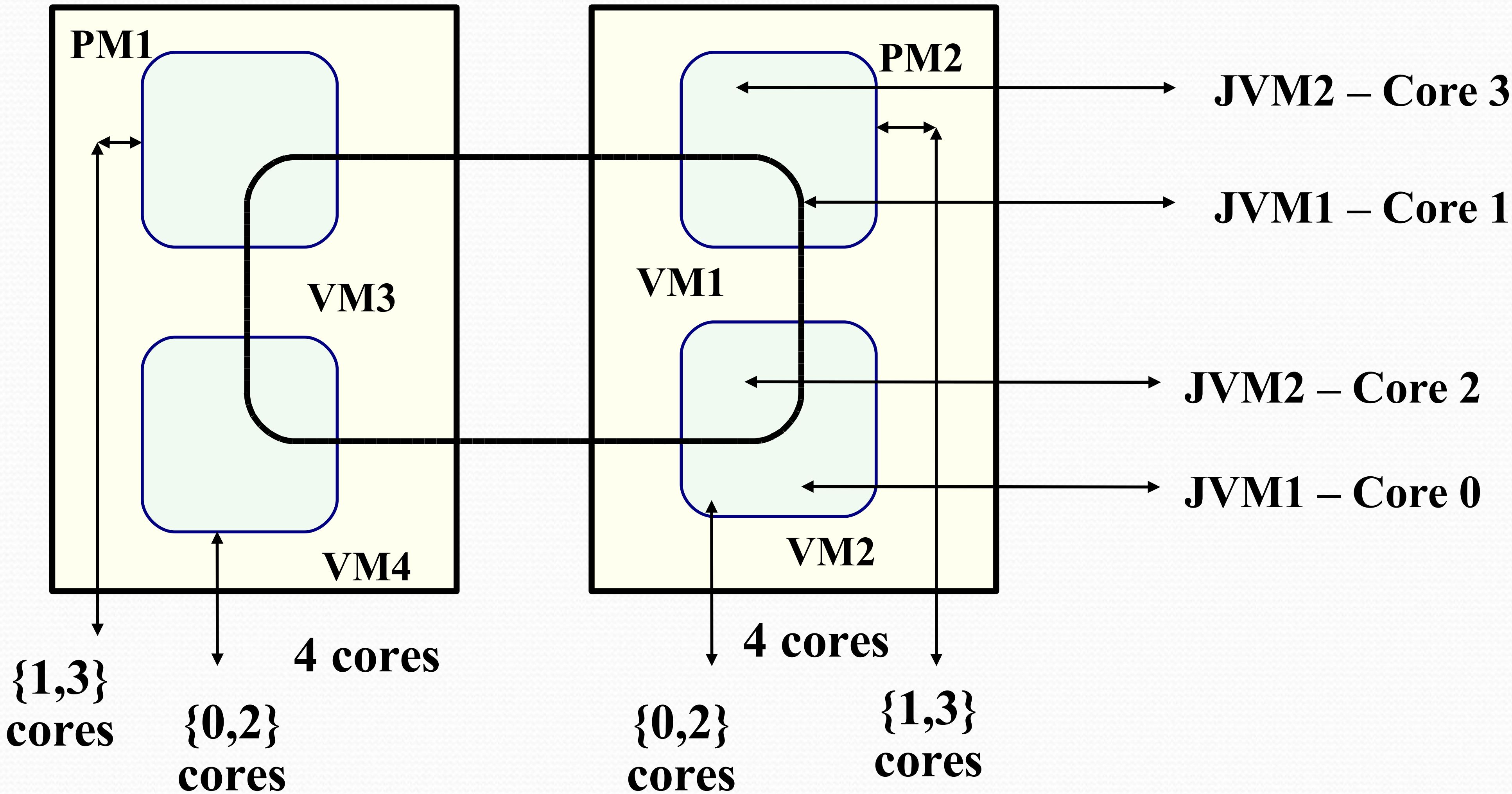
- Java
- Netty Library Used
- Evaluation - YCSB – Yahoo Cloud Serving Benchmark

Hardware/OS Platform

- Nodes on HGCC Cluster
  - Quad-core Xeon 2.53GHz CPU, 8GB memory

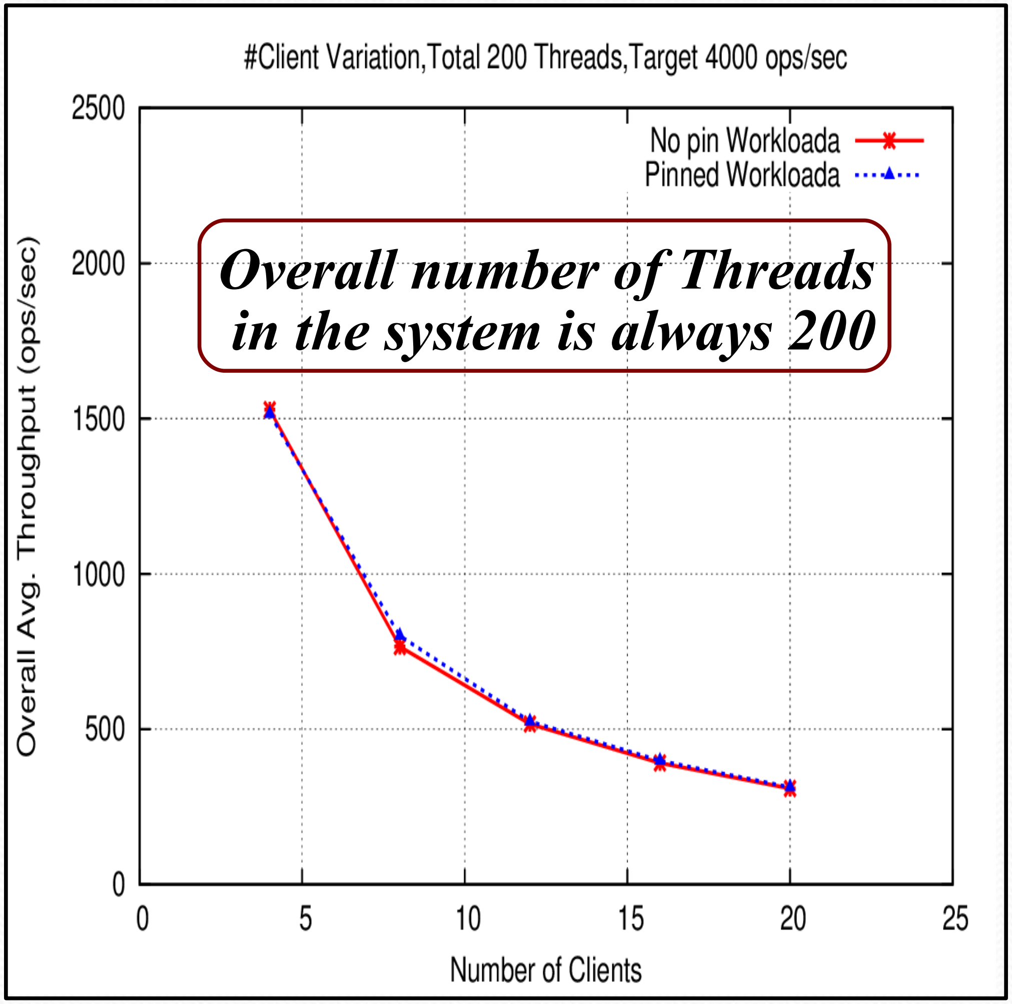
- VMs running with Ubuntu 12.04 32bit with 4 GB memory and 2 vpcus on 2 HGCC nodes

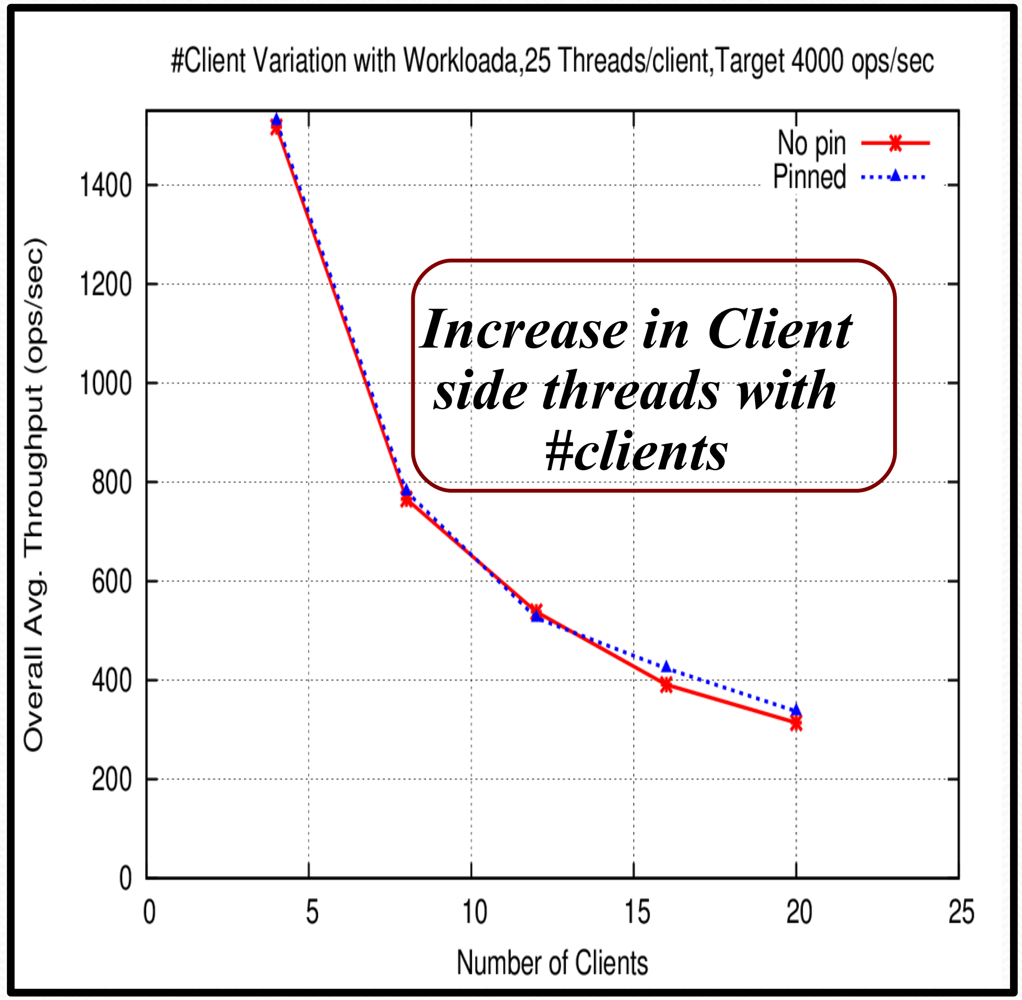
Experimental Set-Up



- 8 Instance Hazelcast Cluster with 4 VMs on 2 PMs
- 2 instances (JVMs) on each VM, pinned to a specific core

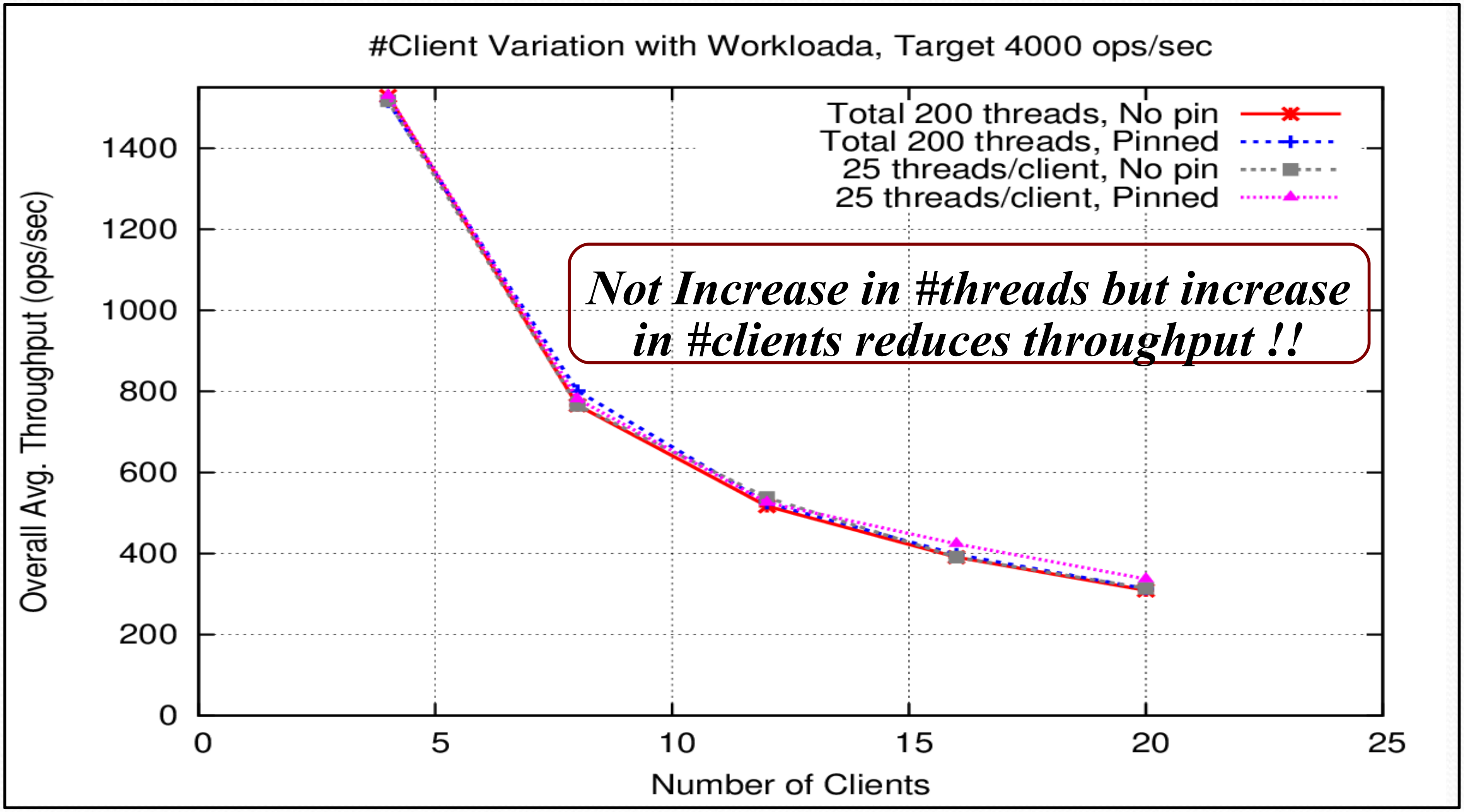
Client Count Variation





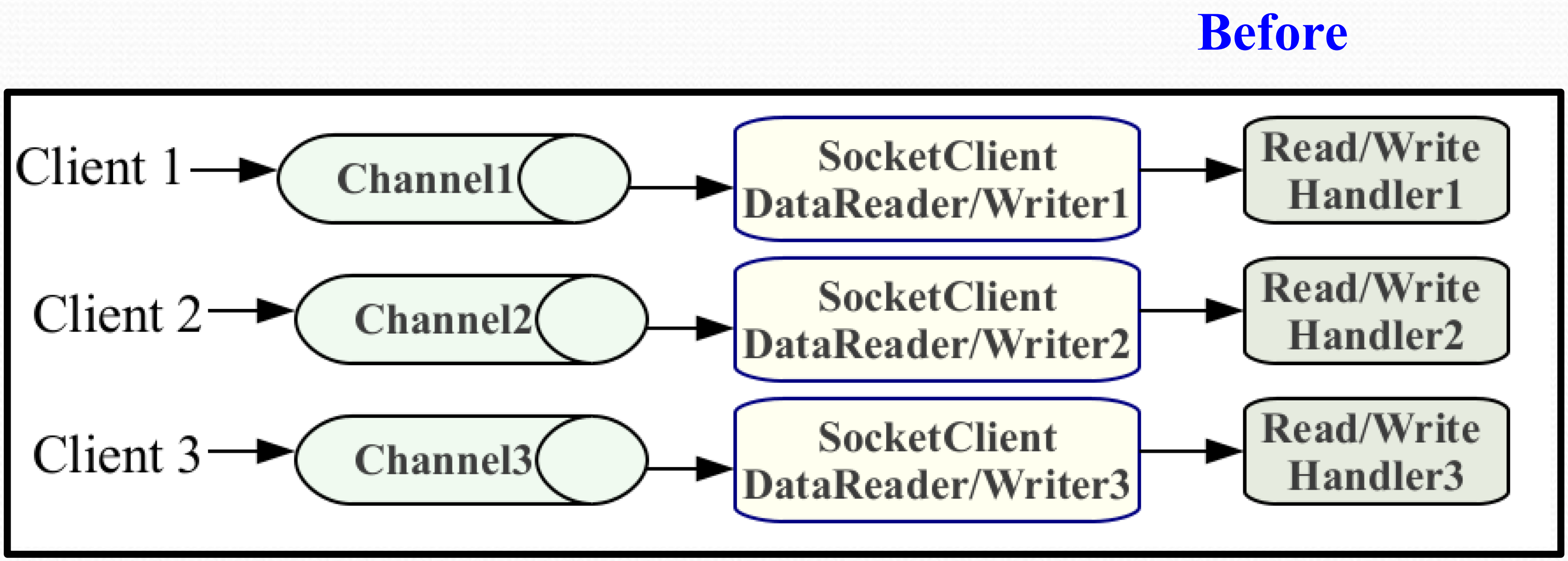
- ***Total number of client side threads used to generate workload fixed to 200 & then varied (increased)***
- ***No difference in performance between with and without pinning***

Client Count Variation

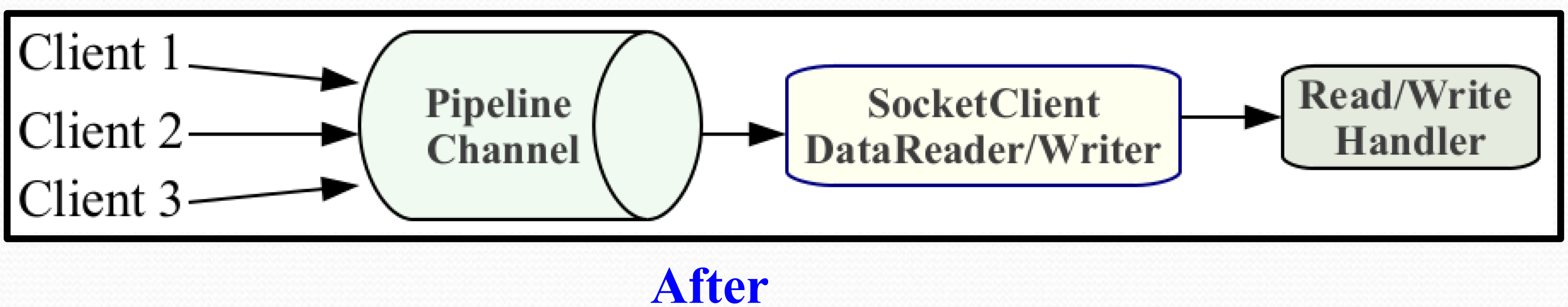


- ***Pinning does not help in performance improvement***
- ***Thread migration/Context switches across cores is not high enough to affect performance***

Multiplexing Socket Channels

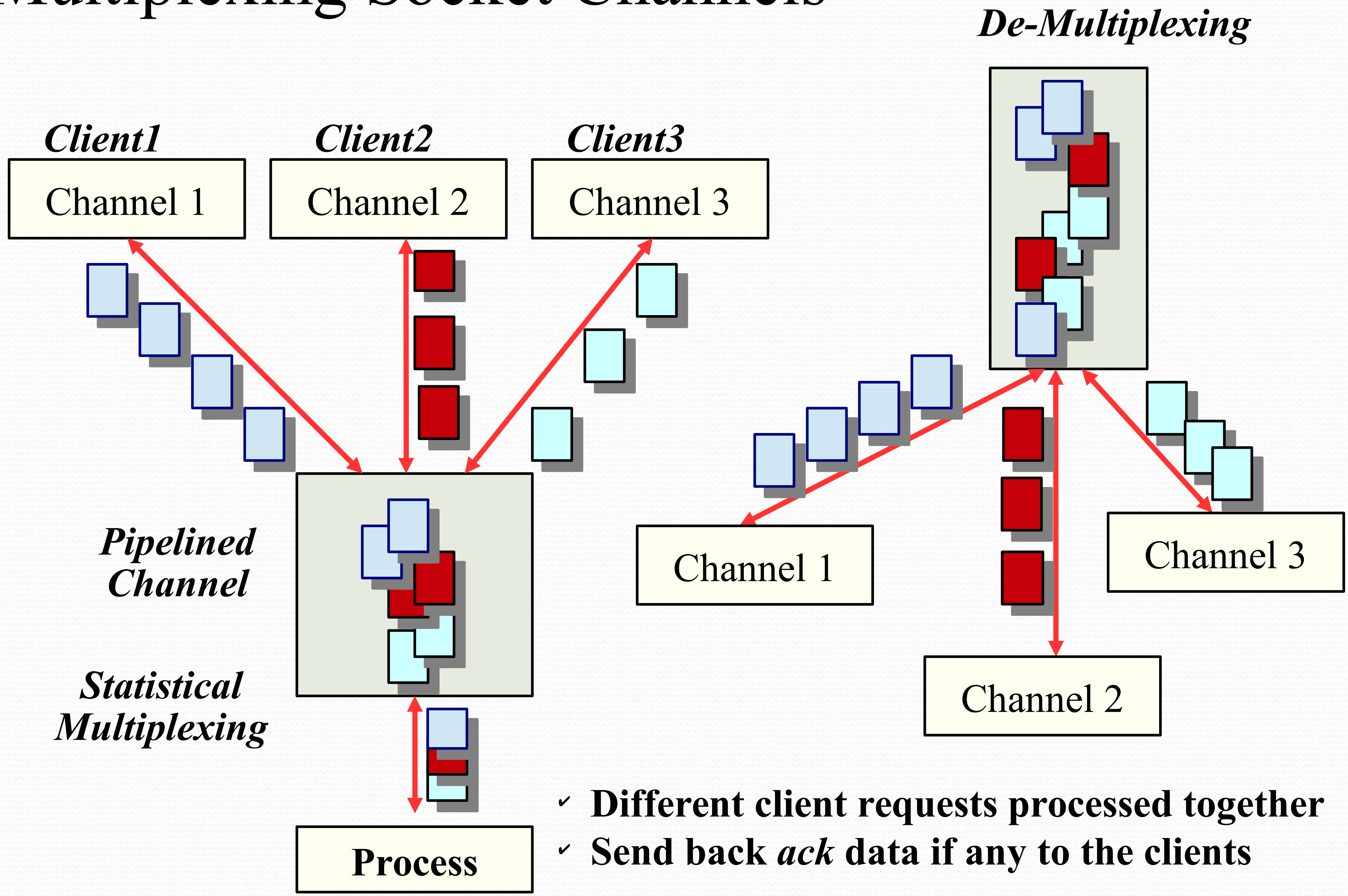


- **Each client has a separate channel, objects/threads created along the way**



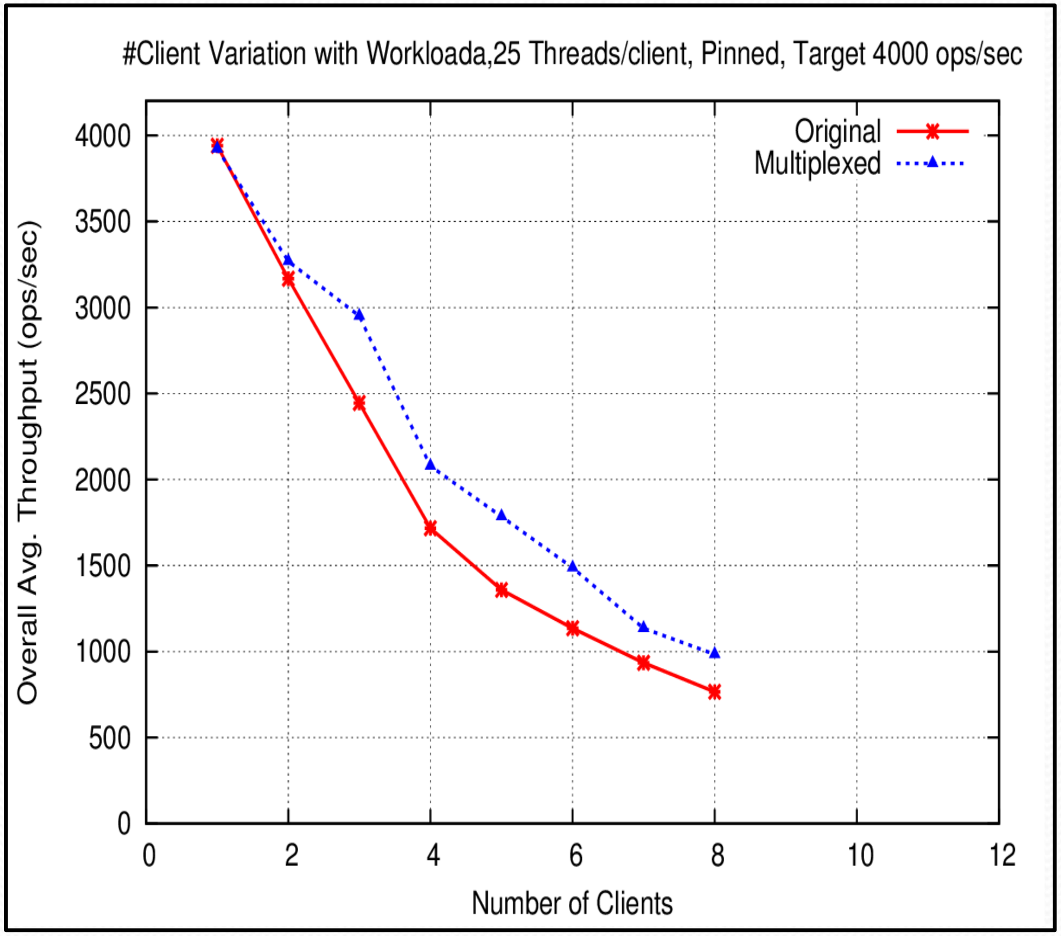
- **Combine the separate channels to a coalesced outbound channel**

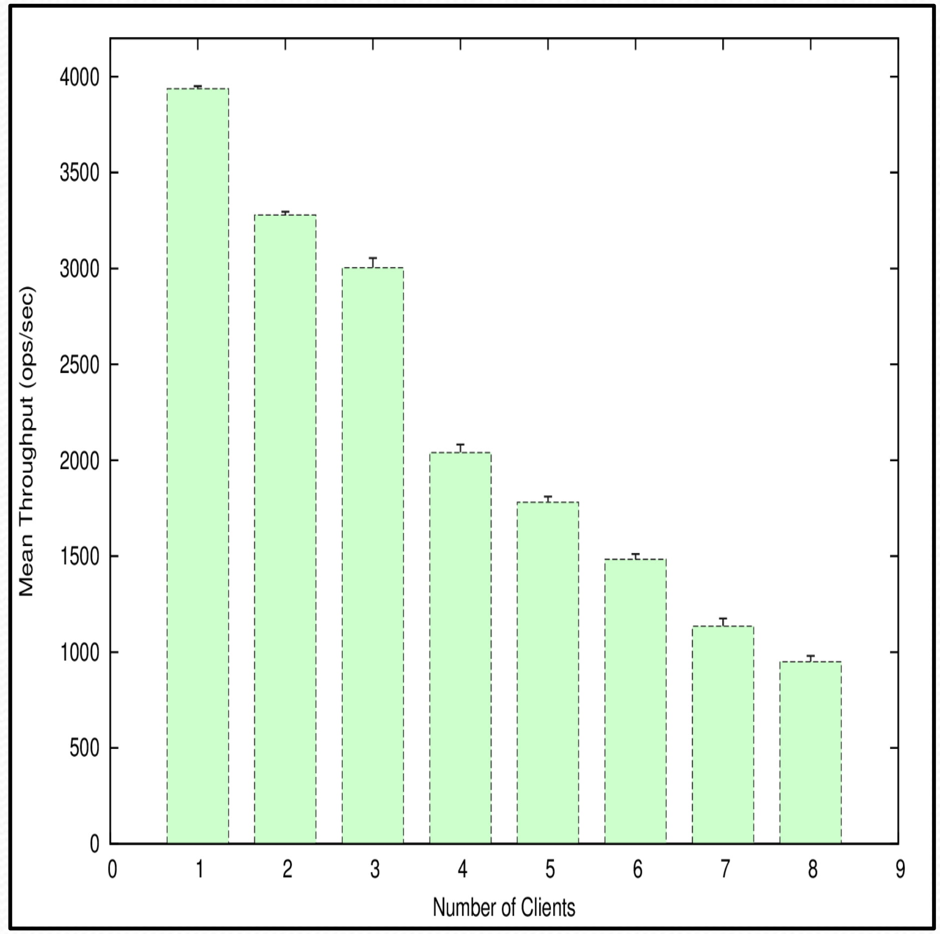
Multiplexing Socket Channels



- **Different client requests processed together**
- **Send back *ack* data if any to the clients**

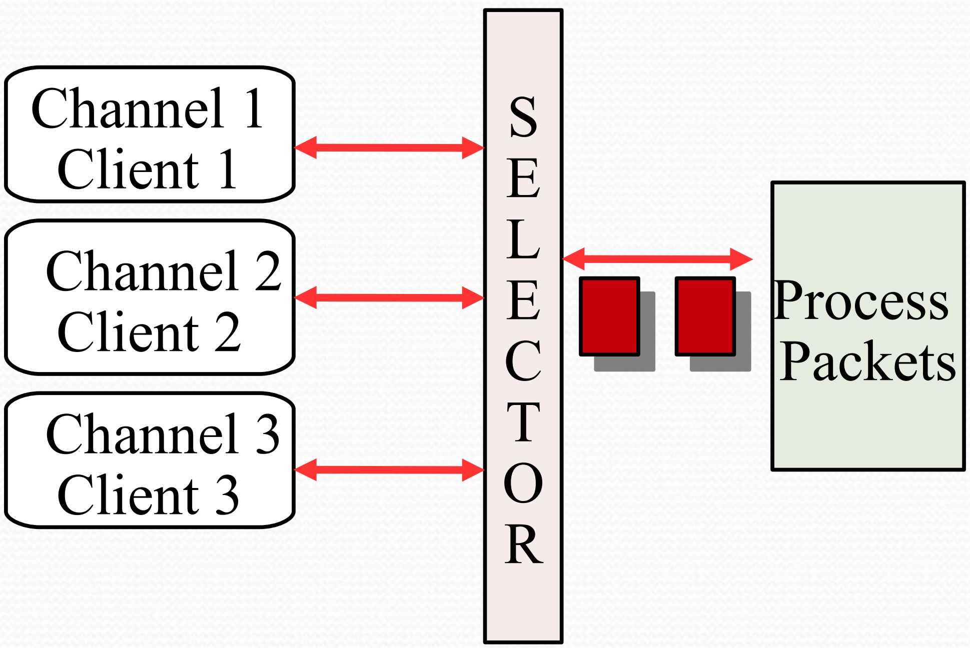
Multiplexing Socket Channels



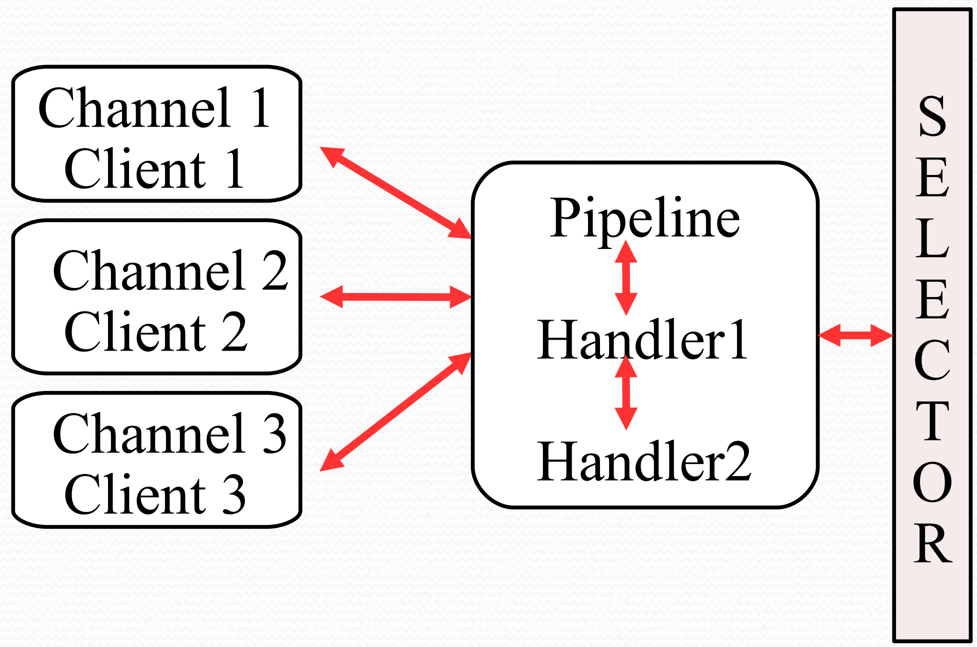


- ***Performance Improvement > 10 % for 5 clients***
- ***Standard Deviation did not exceed 50***
- ***Less endpoint management, better performance***

Multiplexing Socket Channels

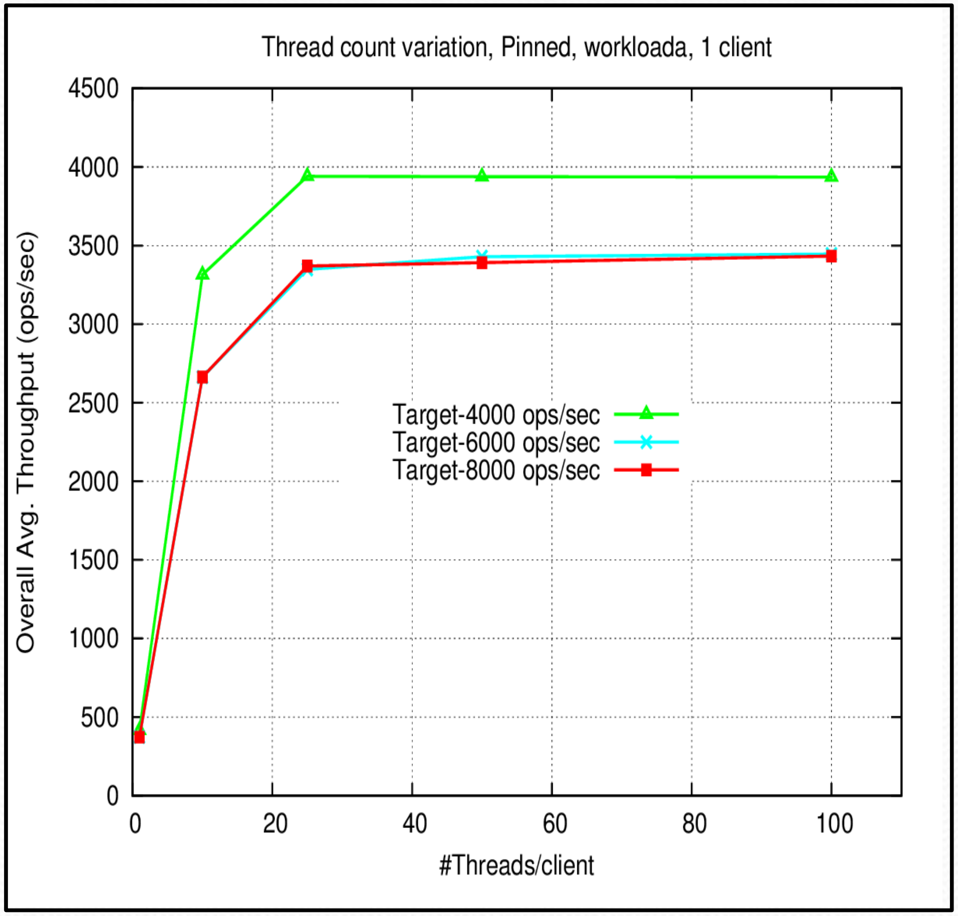


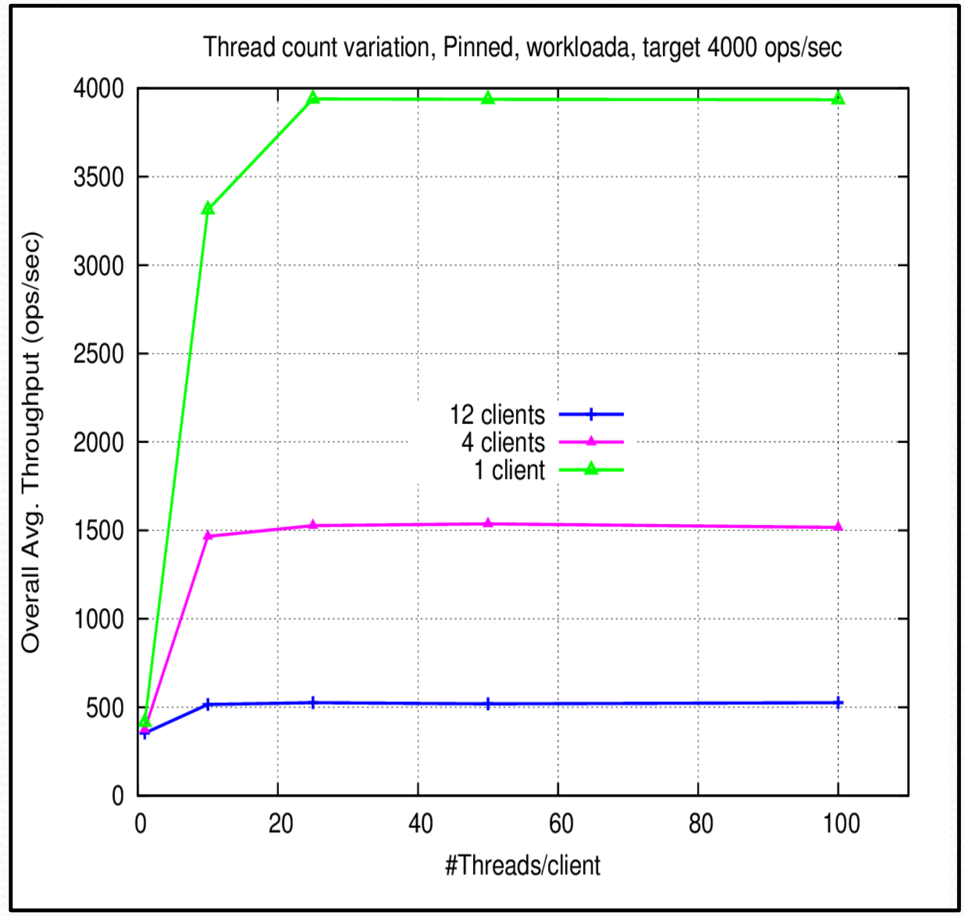
- *Select/Poll*
- *More Context switches*
- *More Runnable Instances - higher threads per request*



- *Less Context switches*
- *Less threads per request*
- *Better performance with concurrency*

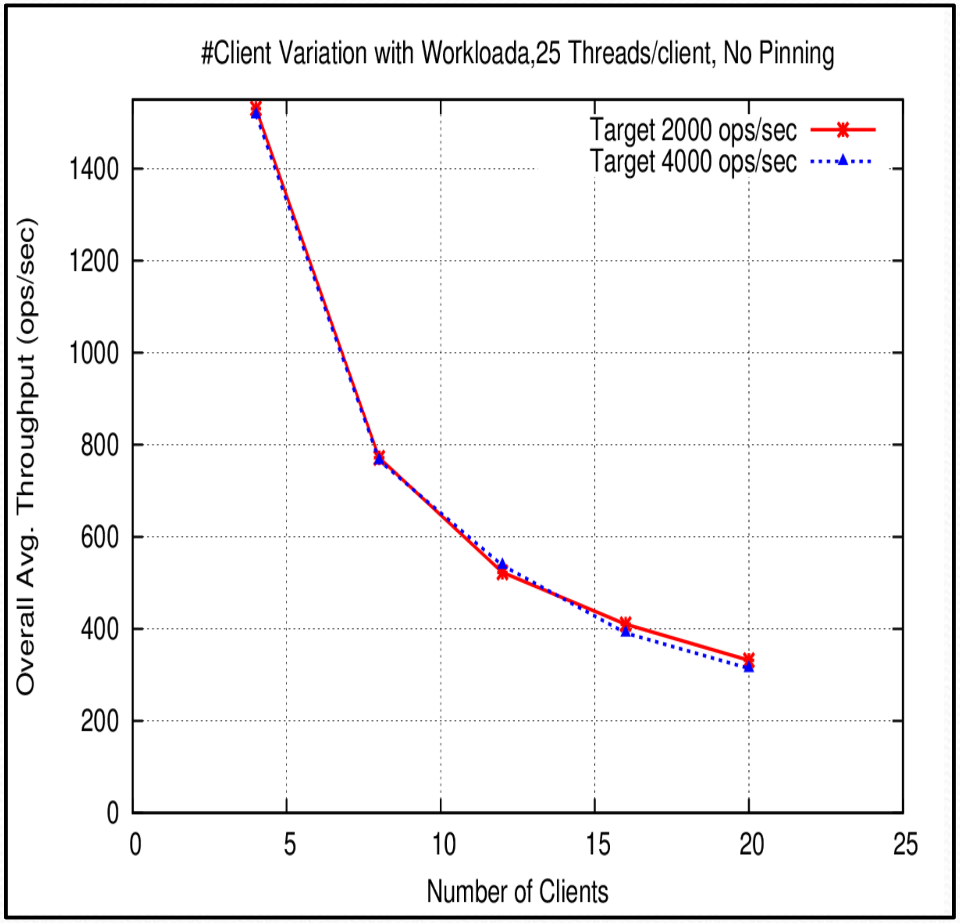
Client Thread Count Variation

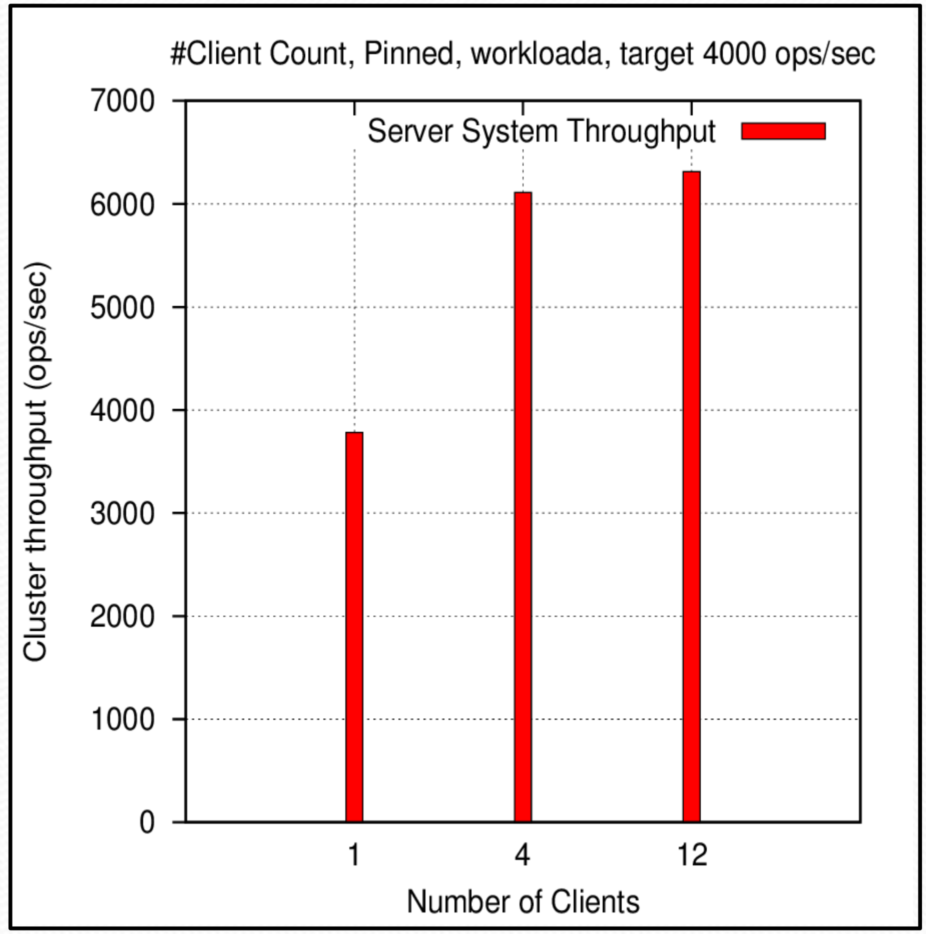




- ✓ ***Single client achieves target, beyond 4000 ops/sec performance drops.***
- ✓ ***With both 4 & 12 clients, system reaches maximum performance level at 25 threads/client beyond which there is no fluctuation.***

Target Throughput





- ***Target doubled, no difference in throughput with the same #clients.***
- ***Cluster throughput from the server's perspective does not exceed 6300 ops/sec.***

Conclusions

- Performance degradation is clear with increasing clients.
- JVM-CPU Pinning does not help in improving performance.
- Beyond a threshold and a specific target, increasing the number of client threads do not affect performance.
- Statistical multiplexing of client socket channels improve overall throughput.

Acknowledgments: Dr. Frank Mueller, Dr. Arun Iyengar, Dr. Helen Gu and NSF for the support.