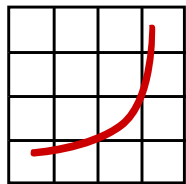




# SPECpower\_ssj2008\*\* Characterization

Anil Kumar, Larry Gray and Harry Li

Intel® Corporation



spec

SIPEW Workshop  
June 28, 2008  
Darmstadt, Germany

\* Other names and brands may be claimed as the property of others

\*\* SPEC and the benchmark names are trademarks of the Standard Performance Evaluation Corporation

Performance Data as of 15 June, 2008.

# Agenda

- **Quick Overview**
  - About the Benchmark
- **Server Resource Usage**
  - Impact of JVM Optimizations
  - General Observations
- **SPECpower\_ssj2008 in “Research Mode”**
  - Workload Generator
  - What If? Experiments
- **Summary**

# **SPECpower\_ssj2008**

## **A Quick overview**

SPECpower\* documentation and results disclosures: [http://www.spec.org/power\\_ssj2008/](http://www.spec.org/power_ssj2008/)

SPECpower\_ssj2008 is intended for “volume server” platforms

# The ssj2008 Workload

ssj = Server Side Java Workload

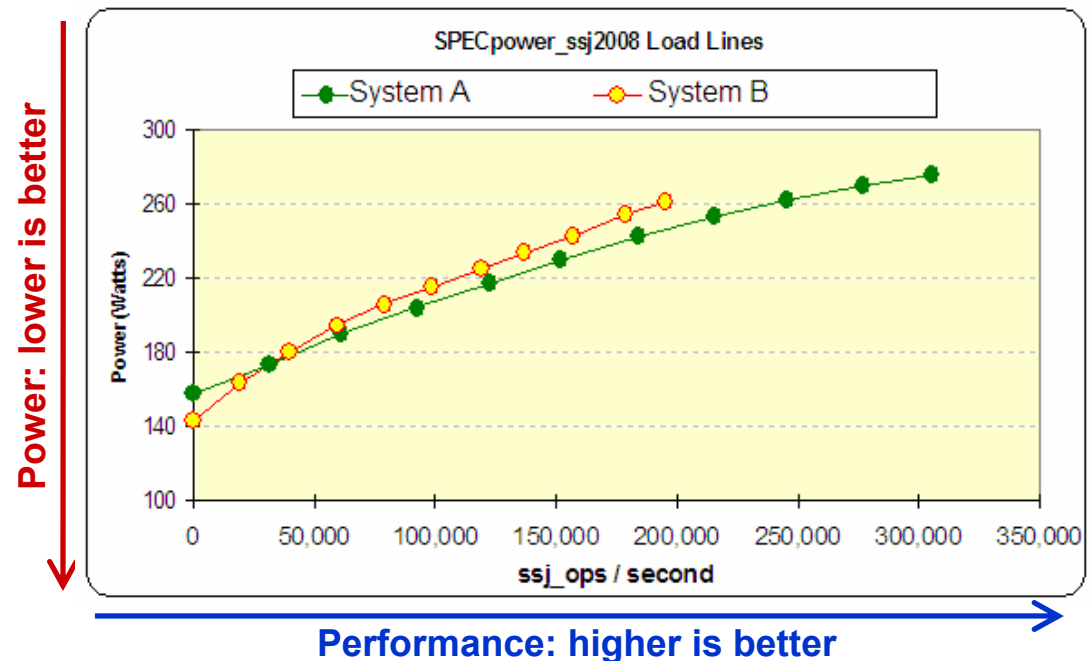
- Based on SPECjbb2005; though *distinctly* different

## Self-Calibrating

- Determines Platform Peak Throughput Capacity

## A “Graduated” Load

- 11 Load Points; Percentages of Peak
- 100%, 90%, 80%, ..., 20%, 10%, zero load (active idle)



**Platform Comparisons With a “Load Line”**



# “Graduating” the Workload

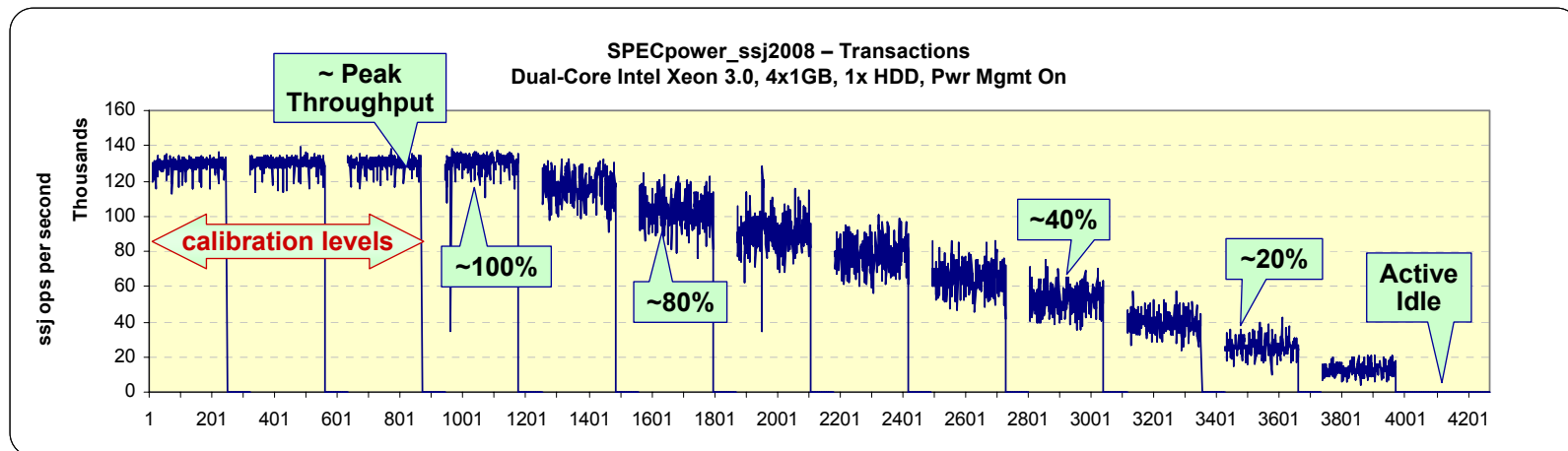
**First:** A Calibration Phase: Run to Peak Transaction Throughput

- # warehouses or threads = # cores,
- scheduling is “ungated”

**Next:** Load Levels: Gradations Based on Calibrated Throughput

- Average of last two calibration levels = “*peak calibrated throughput*”

Actual Average Per Cent of Calibrated Peak Throughput									
99.8%	90.1%	79.6%	69.7%	60.2%	49.9%	40.1%	30.6%	20.0%	9.9%



**Seeks The Peak, Runs and Reports a “Load Line”**

# SPECpower\_ssj2008 - Metric Definition

## The Primary Metric for SPECpower\_ssj2008:

$$\text{overall ssj\_ops/watt} = \sum \text{ssj\_ops @ 11 pts} / \sum \text{avg watts @11 pts}$$

*(includes power at the active idle state)*

Table from SPECpower\_ssj2008 Full Disclosure Report

Performance			Power	Performance to Power Ratio
Target Load	Actual Load	ssj ops	Average Power (W)	
100%	99.10%	220,306	276	799
90%	90.40%	200,860	269	746
80%	79.50%	176,684	261	677
70%	70.30%	156,344	254	616
60%	59.60%	132,525	245	541
50%	49.60%	110,222	237	465
40%	40.20%	89,388	229	390
30%	30.10%	66,875	221	302
20%	19.90%	44,157	213	207
10%	10.20%	22,649	206	110
Active Idle		0	198	0
$\sum \text{ssj\_ops} / \sum \text{power} =$				468

ssj\_ops@100%

ssj\_ops each level

average power each level

performance / power each level

overall ssj\_ops/watt

**Much more data in SPEC report !**

SPECpower\_ssj2008 Intel publication #017

[http://www.spec.org/power\\_ssj2008/results/res2007q4/power\\_ssj2008-20071129-00017.html](http://www.spec.org/power_ssj2008/results/res2007q4/power_ssj2008-20071129-00017.html)



# Initial characterization of SPECpower\_ssj2008

# Hardware and Software

- **SUT: Intel® “White Box”**
  - Dual and Quad Core Intel® Xeon® 2.0 & 3.0 GHz
  - Supermicro\* X7DB8/ Main Board, Super Micro 5000P (Blackford chipset)
  - 4x 2GB FBDIMMs
  - 1x 700W PSU
  - 5U Tower Platform
- **Microsoft\* Windows Server 2003 64 bit**
  - Power Options: Server Balanced Processor Power and Performance
- **JVM:**  
**BEA\* JRockit\* P27.4.0 64 bit**
  - JVM Command Line similar to published results
- **Sampling Rates:**
  - Power: 1 second (average from meter)

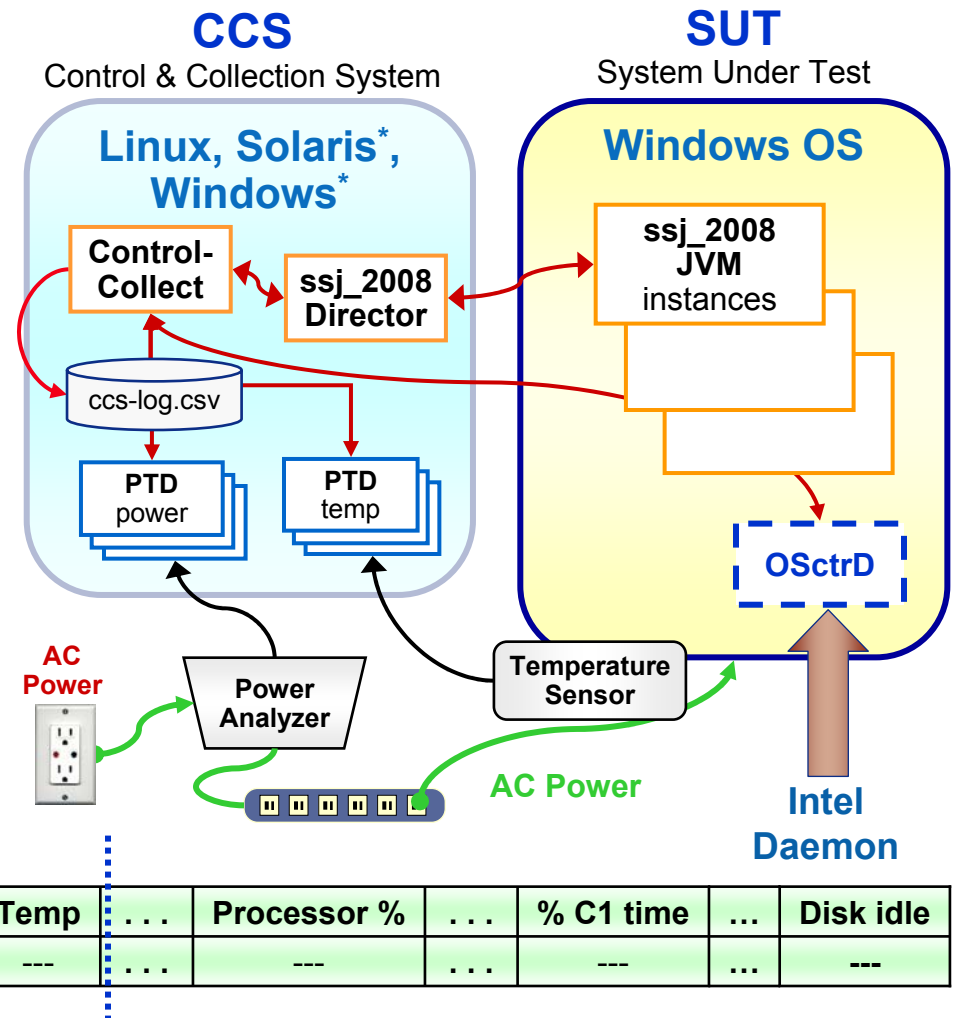
## **SPECpower\_ssj2008**

### **“Research” setup**

- **SSJ Director on SUT**
- **Load level Duration: 120 secs**
- **Delay between Load Levels: 5 secs**

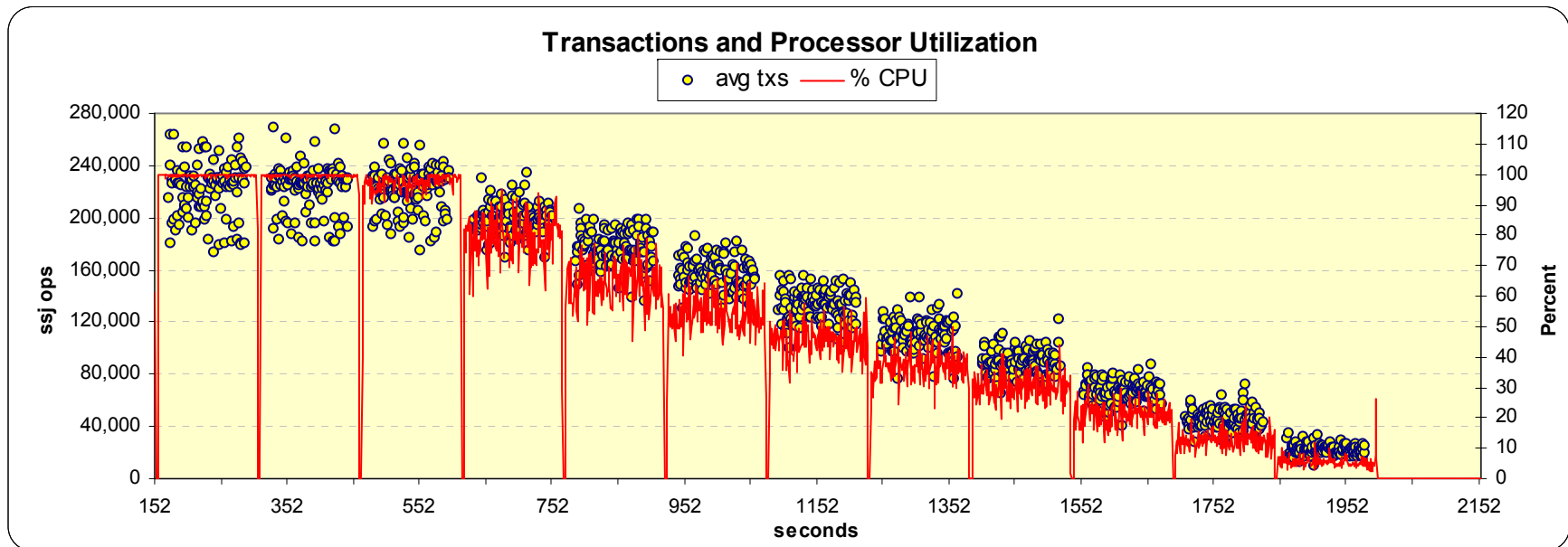
# Collecting OS Counters

- Intel Counters Daemon
  - Counters defined in ccs.props
- Runs on SUT
  - Data to CCS via TCP/IP
  - CCS logs counters *WITH* watts, trans, etc.
- Windows Only
  - Linux port under consideration
- “Integrated” Log
  - Primary advantage



# Transactions (ssj\_ops)

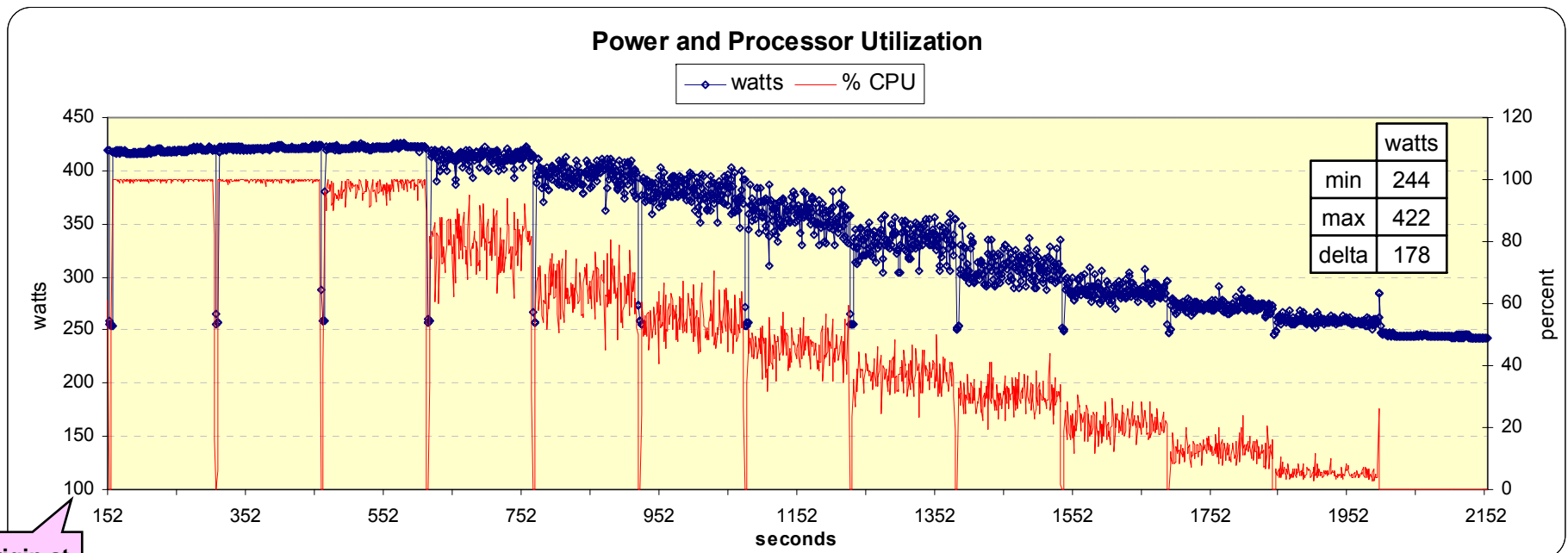
- Average ops per level tracks as expected
  - Per second rate showing intended variability within load level
    - Negative Exponential Inter-Arrival Time batch scheduling
- CPU % tracks load
  - As expected on Intel Core 2 architecture
  - Other architectures May vary (with/without SMT etc.)



Charts start at Calibration level 2

# Power and Processor Utilization

- Power consumption tracks processor utilization
  - Other components change very little
  - With this and other loads
- **CPU utilization?** → *Not part of the benchmark*

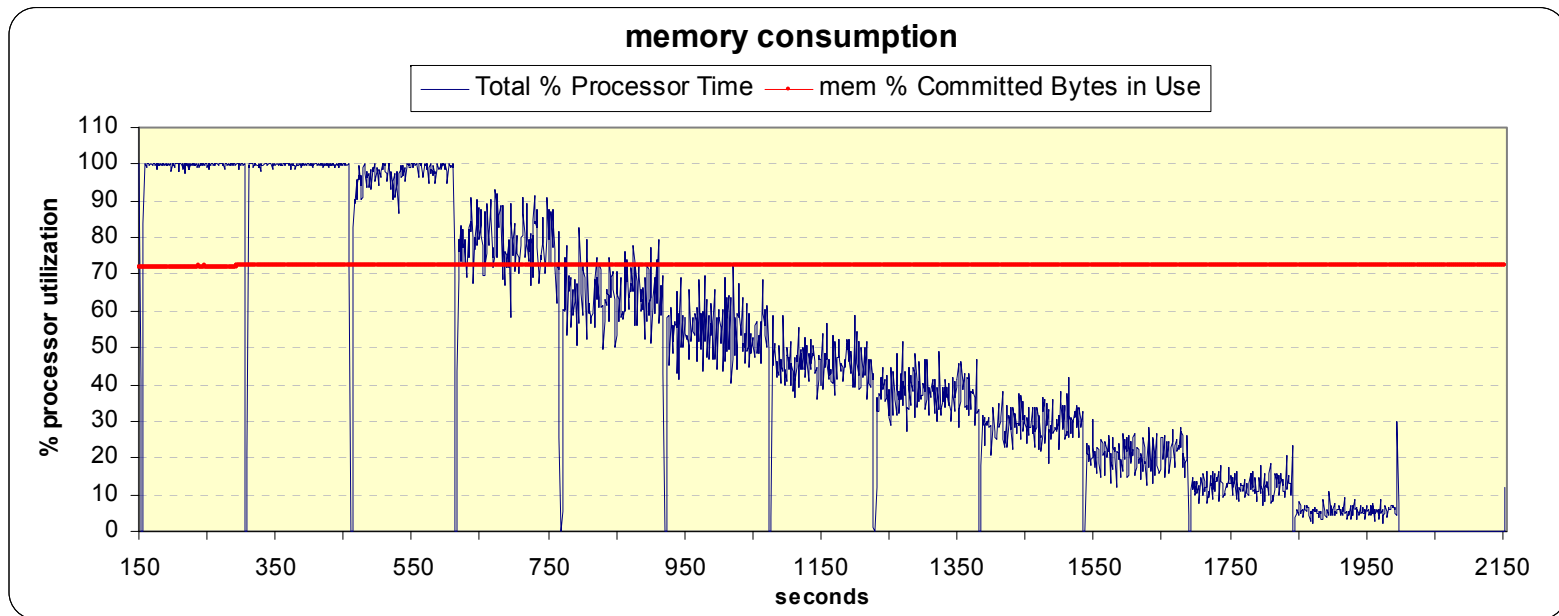


Origin at  
100 watts

**Power Consumption Varies With Load**

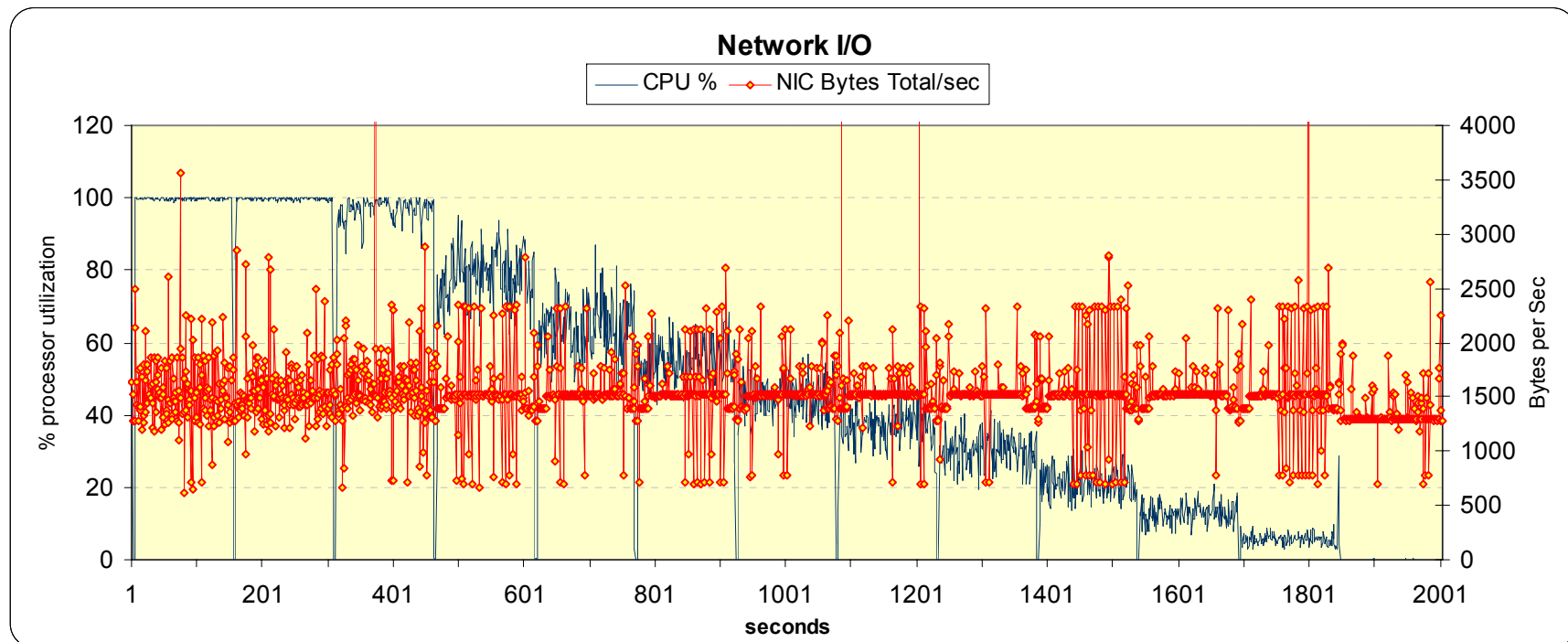
# Memory Utilization

- Constant committed memory in use at all load levels
  - including active idle – JVM(s) still active
- Java heap size remains same throughout the run
  - With typical tuning; `-Xmx==Xms`
- JVM Options dictate memory usage profile



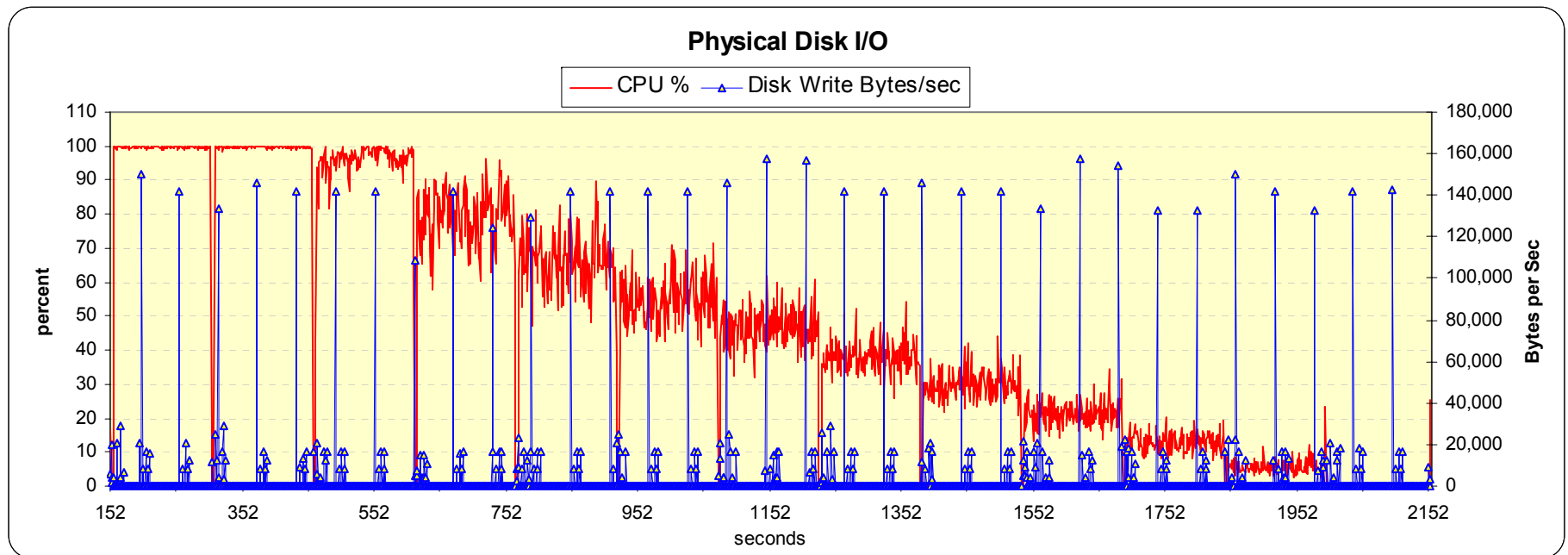
# Network I/O

- ~1.5 K Bytes/sec of network I/O at all load levels
  - including active idle:
- Network I/O from per sec request/response
  - between Control & Collect (CCS) and SSJ\_2008 Director
- Network I/O dependent on setup and configuration



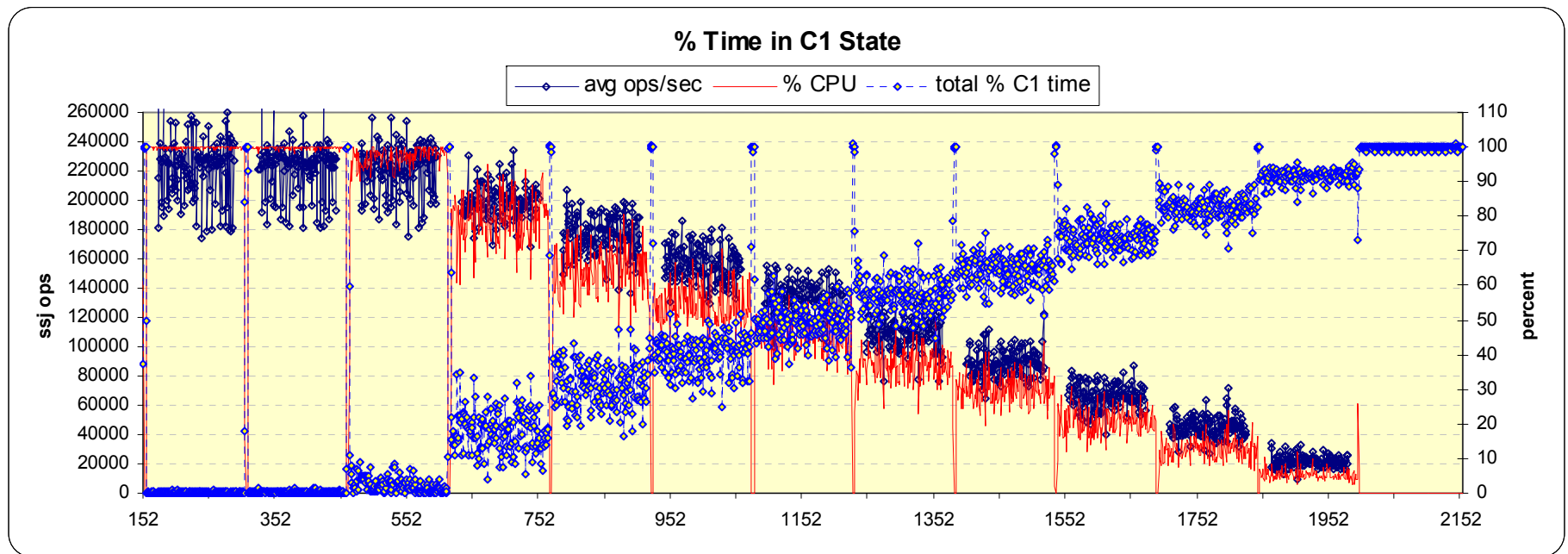
# Physical Disk I/O

- Regular write bursts of ~140K Bytes
  - ~3.3K Bytes/sec writes average for all load levels
  - Most disk writes related to SSJ\_2008 logging
- Reads average zero
- Low I/O rates enable transaction scaling with # cores



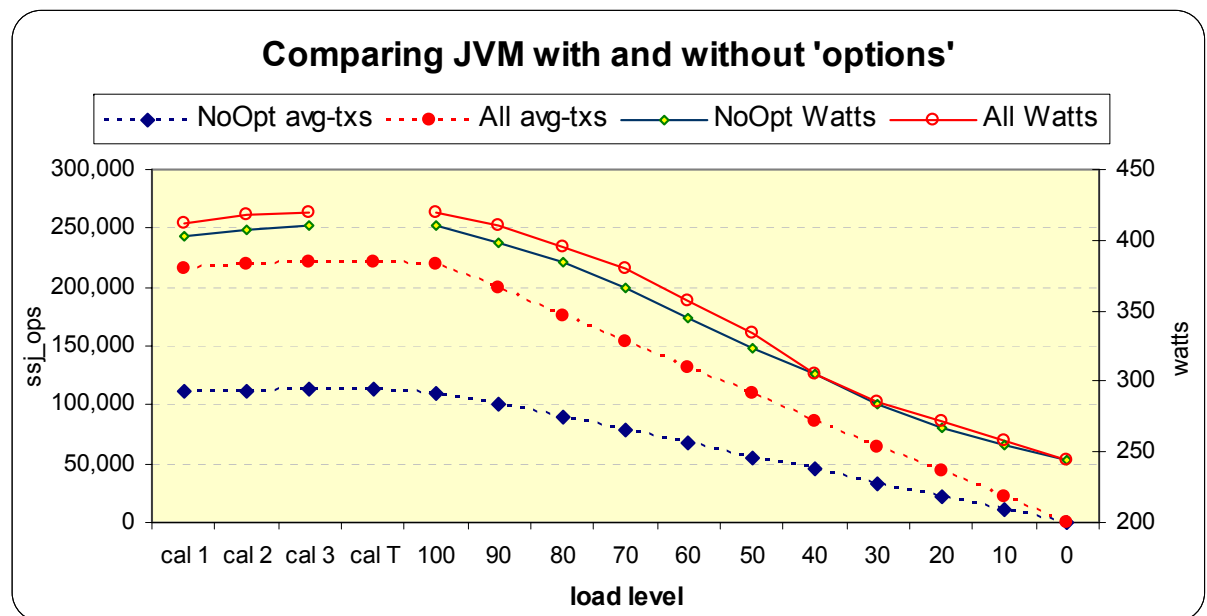
# C1 state

- % Time in C1 State – Inverse of CPU %
  - C1/C1E Time contributes to power saving
- Intel EIST and C1E “enabled” in BIOS
- Varies with Architecture, OS and Power Policies




# Impact of JVM Optimizations

- Experiments with JVM Options
  - JAVAOPTIONS\_SSJ="" (None, default heap and optimization)
  - JAVAOPTIONS\_SSJ="-Xms3000m -Xmx3000m -Xns2400m -XXaggressive -XXlargePages -XXthroughputCompaction -XXcallprofiling -XXlazyUnlocking -Xgc:genpar -XXtlasize:min=12k,preferred=1024k"
- Performance Loss ~50%
- Power Less by 0 to 3% less
- Your Results dependent on JVM and options



# General Observations

- Processor Intensive Workload
  - Graduation (% of peak) well behaved, <2% from target.
  - Processor utilization follows the load line (architecture dependent)
- Memory % Committed – constant across load line
  - Behavior is JVM options and JVM dependent
- % Time in C1 State → Inverse of CPU %
  - C1 Transitions per second highest at idle
- Disk I/O – Regular bursts of ~140K byte writes
  - ~3.3K bytes/sec for all load levels
- Network I/O - ~1.5K Bytes/sec, ~constant across load line
- Software stack – well tuned - is critical to performance



# **SPECpower\_ssj2008**

**as a**

# **Workload Generator**

# Using the “kit” in “Research Mode”

- “The Benchmark” Requires *Fixed* Parameters.
  - **Strict Rules** for Publishing on SPEC web site
  - “Compliant” runs follow run rules
- The Benchmark “kit” Can Serve Other Roles
  - Experiments, Studies, Behavior Analysis, etc.
- Can Change Behavior by Modifying Properties
- **Measure the Power Impact of:**
  - Two Transaction mixes
  - Varied batch sizes
  - Alternate Queuing Methods
  - Adding threads

**For ssj2008 1.0; 35 Properties to Change Behavior**

# Different transaction mix

FIXED quantity of work over FIXED time interval arrives at Server,  
with Negative Exponential inter-arrival time (Poisson distribution):



A batch has 1000 transactions consisting of SIX different types with fixed probability distribution of types.

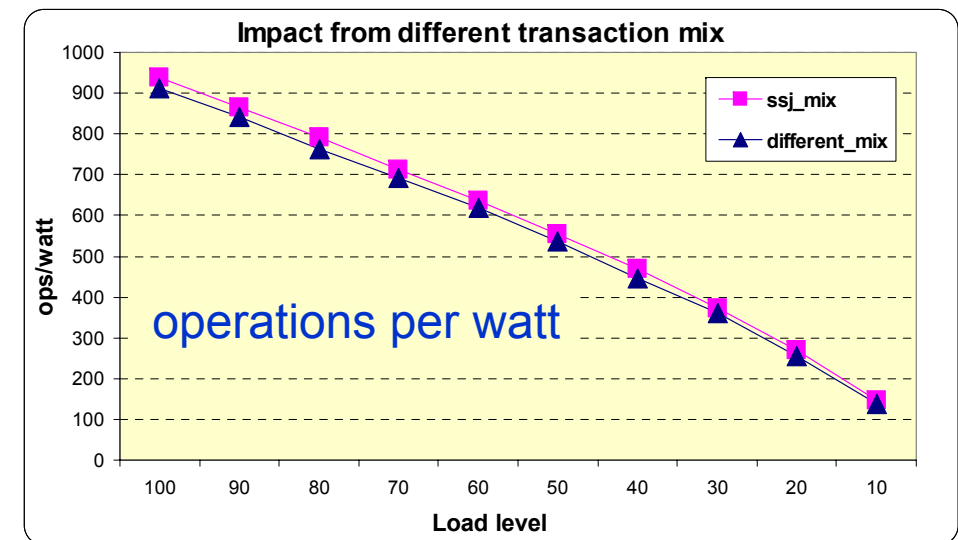
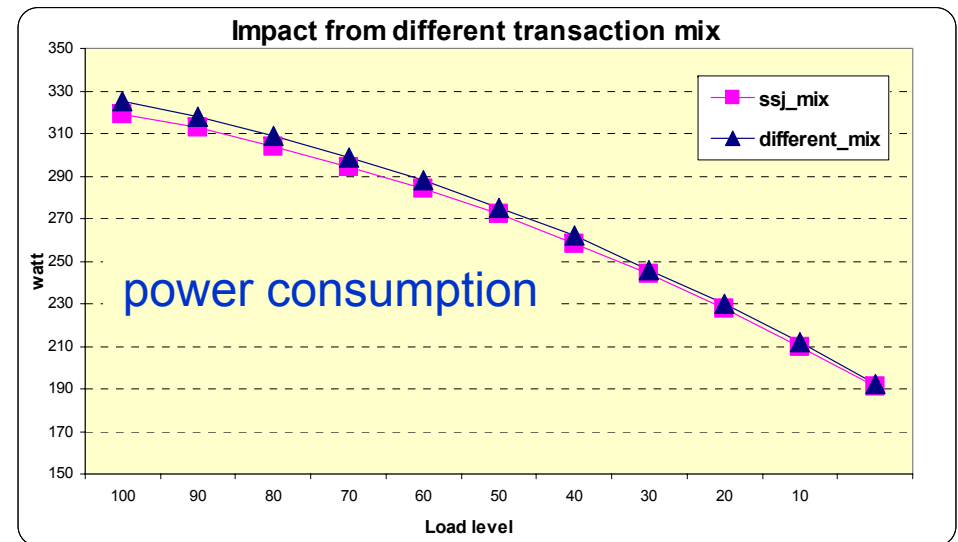
	<u>SSJ MIX</u>	<u>Alternate MIX</u>
<b>New_order</b>	<b>10</b>	<b>15</b>
Payment	10	10
Order_status	1	1
Delivery	1	1
Stock_level	1	1
<b>Cust_report</b>	<b>10</b>	<b>5</b>

(Experiment 1)      (Experiment 2)

□ What is the impact on power consumption?

# Impact of different transaction mix

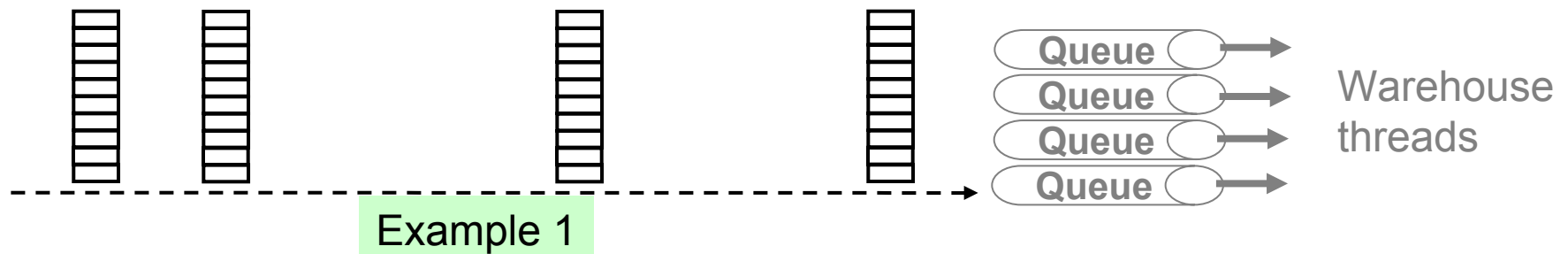
- Different transaction mixes change cache and/or memory access and utilization
- Power consumption only slightly different
- Ops/watt reflects change in power with the alternate transaction mix
  - Different work from 50% more “new order” transactions



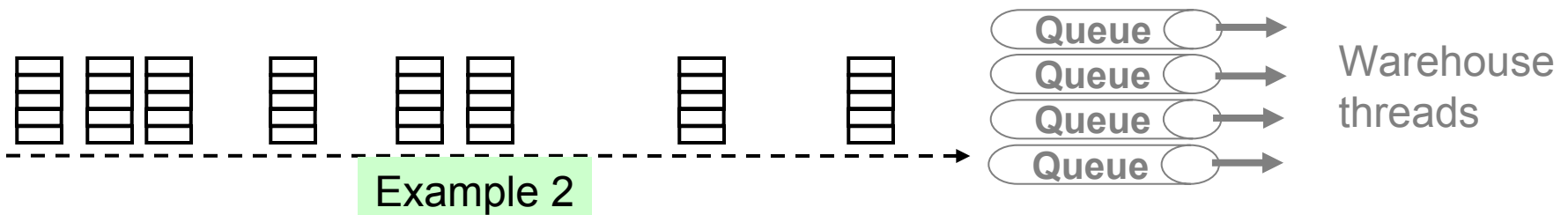
# Batch sizes and Arrival Time

**FIXED quantity of work over FIXED time interval arrives at server:**

Large batch size = less frequent arrival of bigger tasks



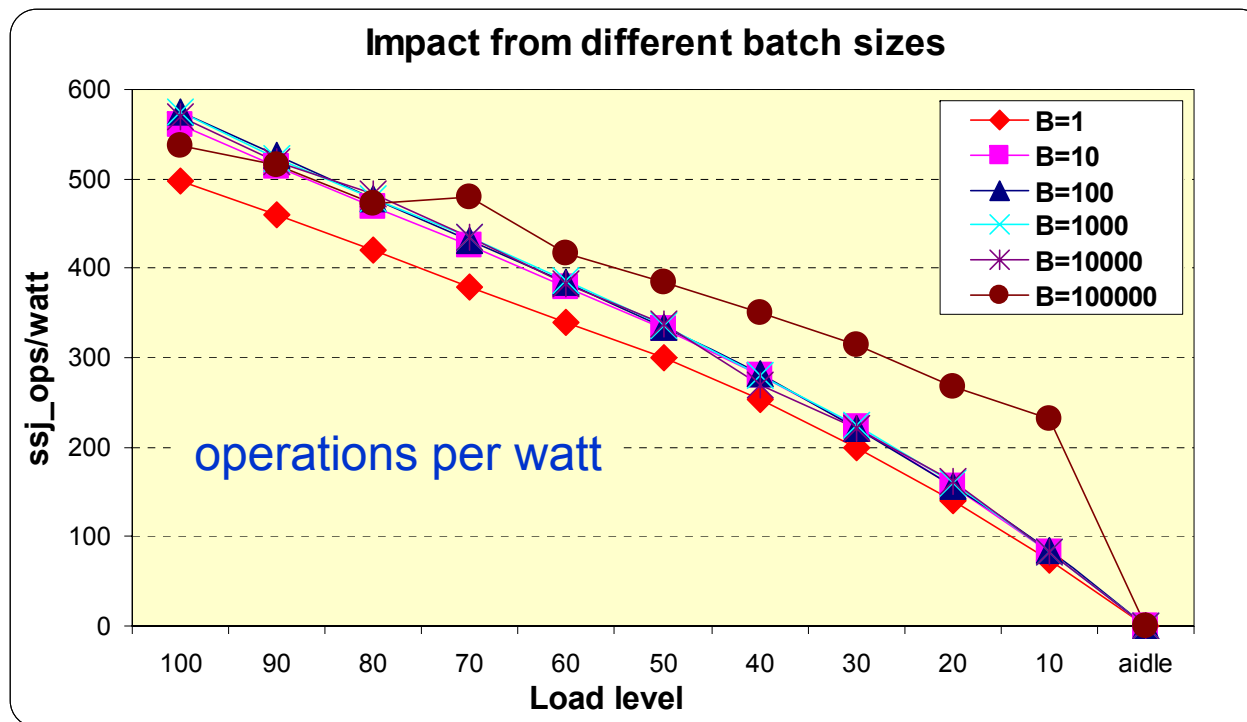
Small batch sizes = more frequent arrival of smaller tasks



□ What is the impact on power consumption?

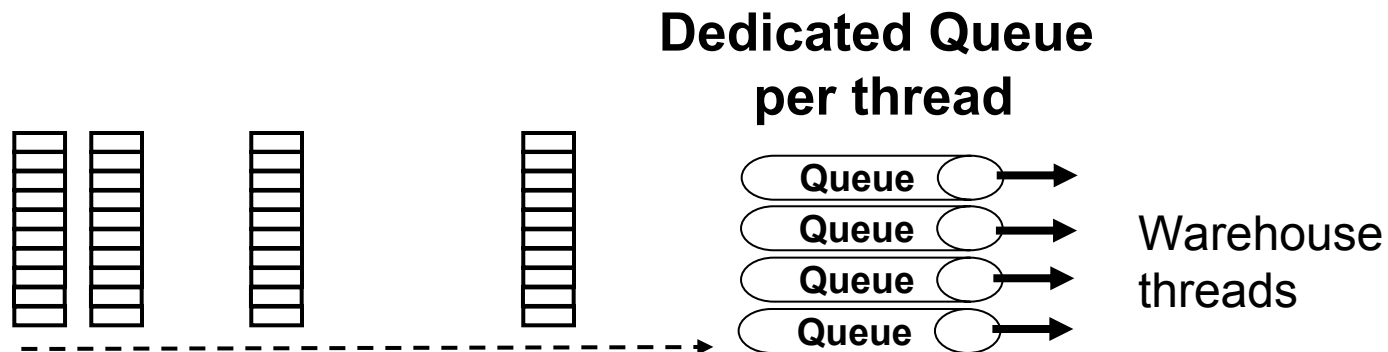
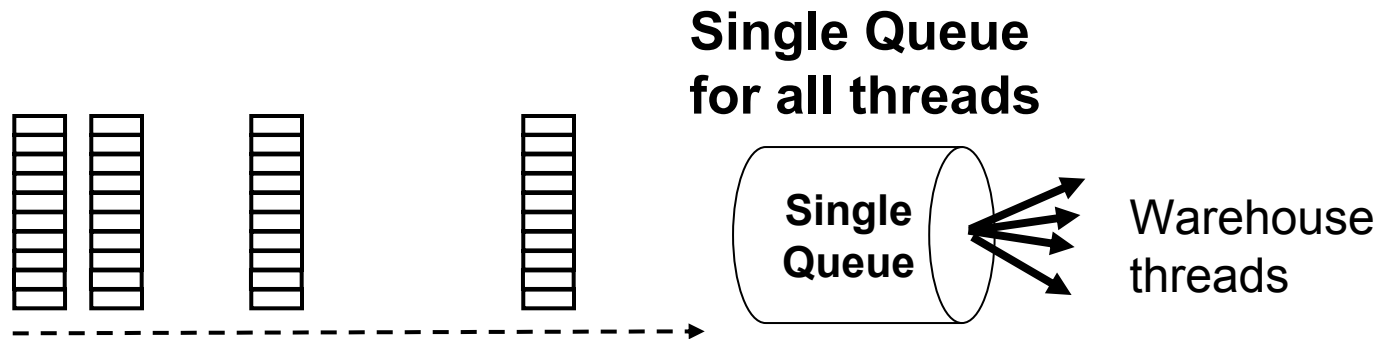
# Impact of different batch sizes

- Default batch size is 1000 transactions
- Experiment with batch sizes from 1 to 100K



- More power saving opportunities with large batch sizes

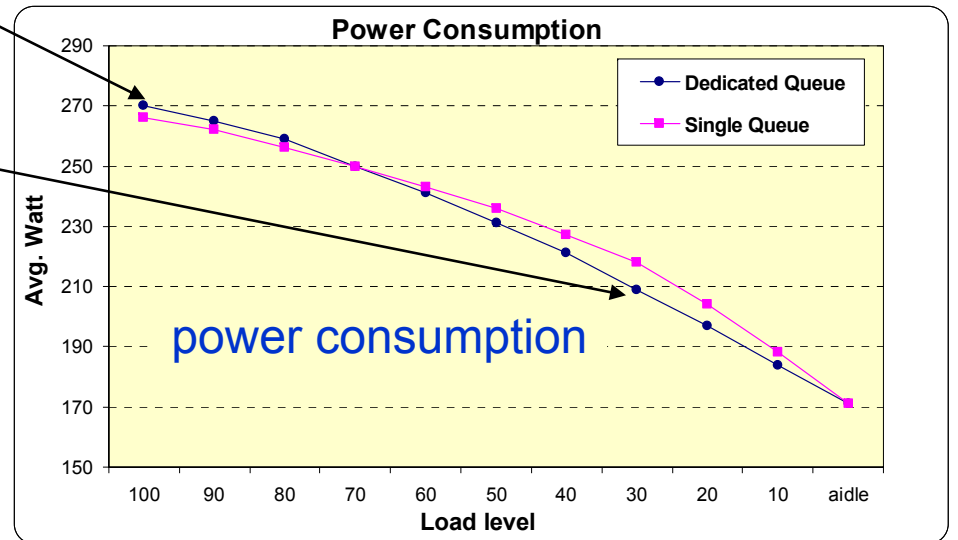
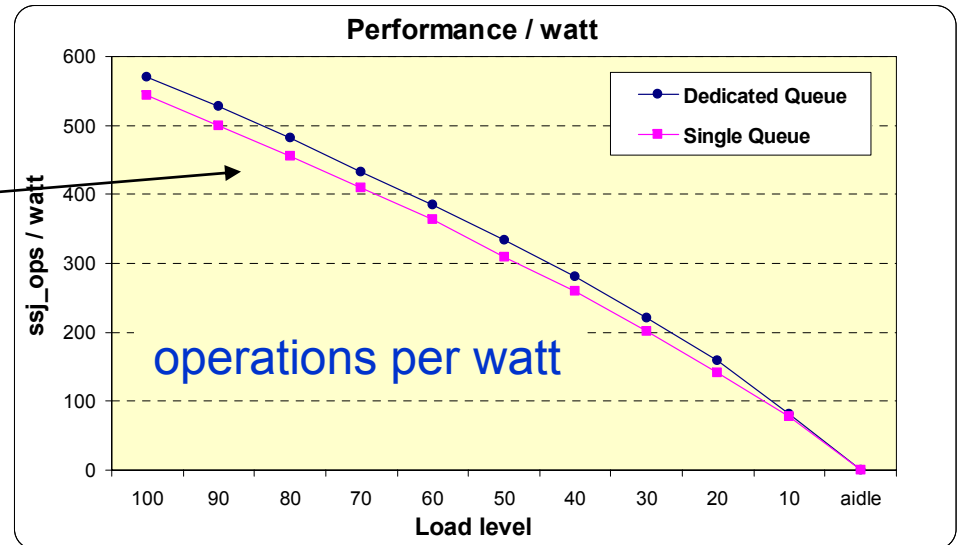
# Alternate Queuing Methods



□ What is the impact on power consumption?

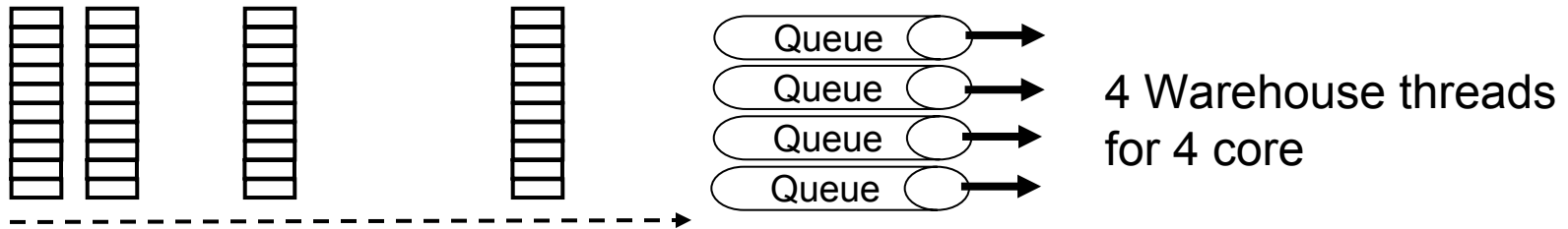
# Alternate Queuing Methods

- Dedicated queue:
  - Better ops/watt at all load levels due to less contention
  - Higher avg watts due to higher ops@100%
  - At lower load levels, similar ssj\_ops; lower avg watts from less contention
- Better approach to this experiment would be:
  - use targeted (fixed) throughput

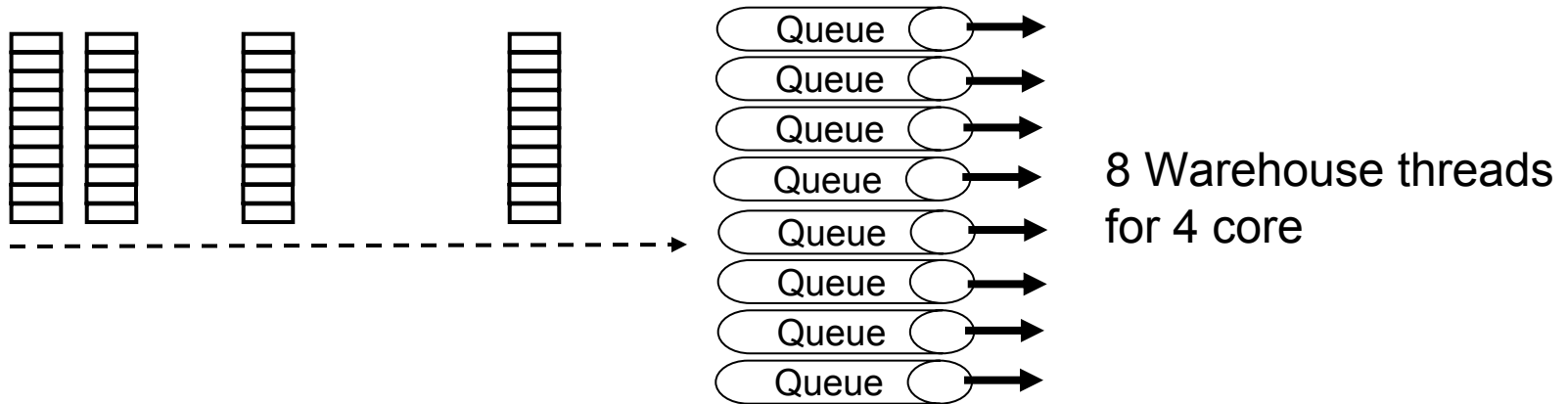


# Threads per core

## 1x threads per core



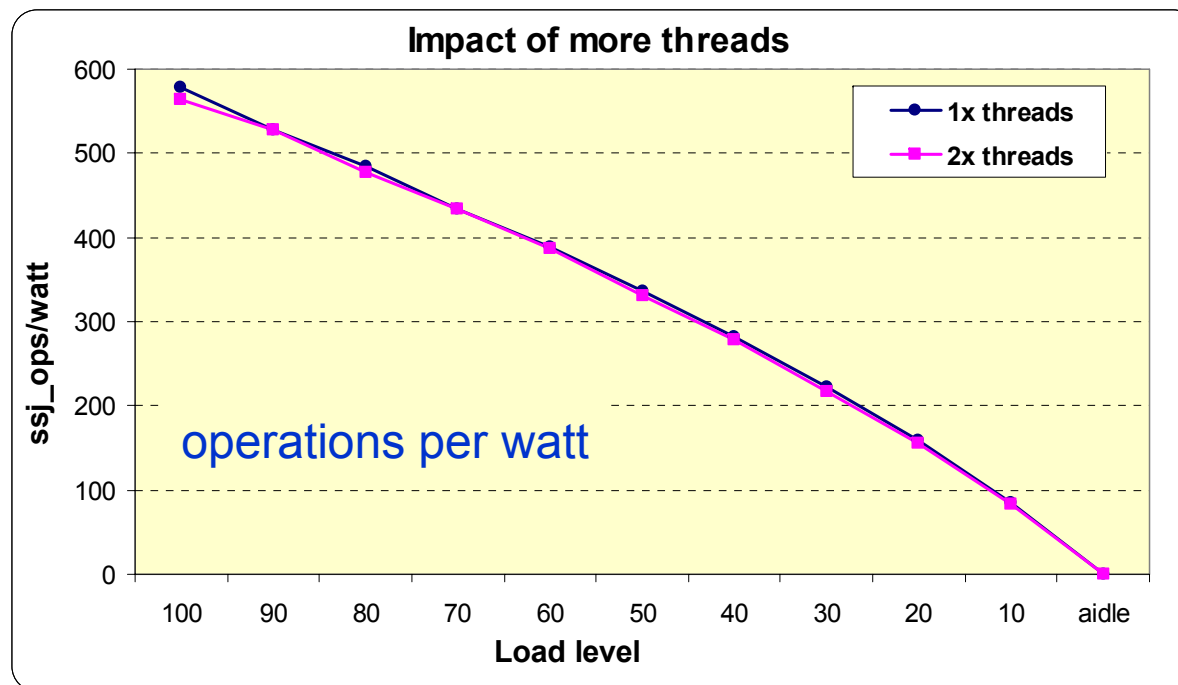
## 2x threads per core



□ What is the impact on power consumption?

# Impact of Threads Per Core

- Assumption: Two threads per core will increase contention, context switching, cache and memory pressure, etc.



- Surprise: Minor difference between 1x and 2x threads/core
- Above is Architecture Dependent: >3-5x could be interesting

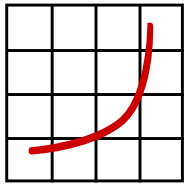
# Other Potential Experiments

- Change duration of load level (e.g. 120 seconds)
  - `input.load_level.length_second = 240`
- When changing hardware behavior (example: Power Management) use “Target” throughput:
  - `input.load_level.target_max_throughput = #`
- Change load level: order, limit, or mix
  - `input.load_level.percentage_sequence = # # # #`
  - `input.load_level.throughput_sequence = # # # #`
- Change data footprint
  - `input.warehouse_population = 60`
  - `input.override_itemtable_size = 20000`
- Logs gory details of transaction behavior
  - `input.scheduler.log_arrival_rates = true`

**SPECpower\_ssj\_EXPERT.props has ~35 “behavior changing” properties**

# Summary

- Interesting New and Different Benchmark
- **AND** Workload Generator
- SPECpower FDRs contain unprecedented amount of data
  - [http://www.spec.org/power\\_ssj2008/](http://www.spec.org/power_ssj2008/)
- Many, Many more experiments and studies “to do”.
- Our Hypothesis: At the same system load (% processor utilization), other similar workloads will show similar power consumption
  - On the same platform and configuration
- **For More Information; Design Docs on SPEC web site**



spec