#SHAREorg

SHARE
Technology · Connections · Results

Performance Architecture and Platform Positioning

Joe Temple (jliitemp@us.ibm.com)
IBM

August 6, 2012
11847

SHARE in Anaheim 2012

Common Metrics

- There have been many attempts to create “a common metric” for the relative capacity of servers
- There are even vendors such as Ideas International (RPE) and IDC (QPI) who sell their version for various machines
- Many people in the Intel community still rely on “MHz” or interpolate from published SPECint Rate results
- Many UNIX people particularly in the AP countries who use a metric estimated or interpolated from published TPC-C values.

The Problem

- Which metric is right?
  - They show a wide variance
  - Switching metrics can “reorder the list”
  - Interpolation between measured or published results leads to even more variation
- A single metric will not “cover” all workloads
  - Relative Capacity is workload dependent
- Little or no understanding of the differences between the machines is conveyed by the metric values.
- An accepted metric drives machines toward a common denominator and loses its effectiveness as a differentiator over time.
  - Administratively declaring an official metric is like declaring that pi = 3 “because it makes the math easier”

The Result

If your understanding of Relative Capacity and server performance is based on a “Common Metric” …

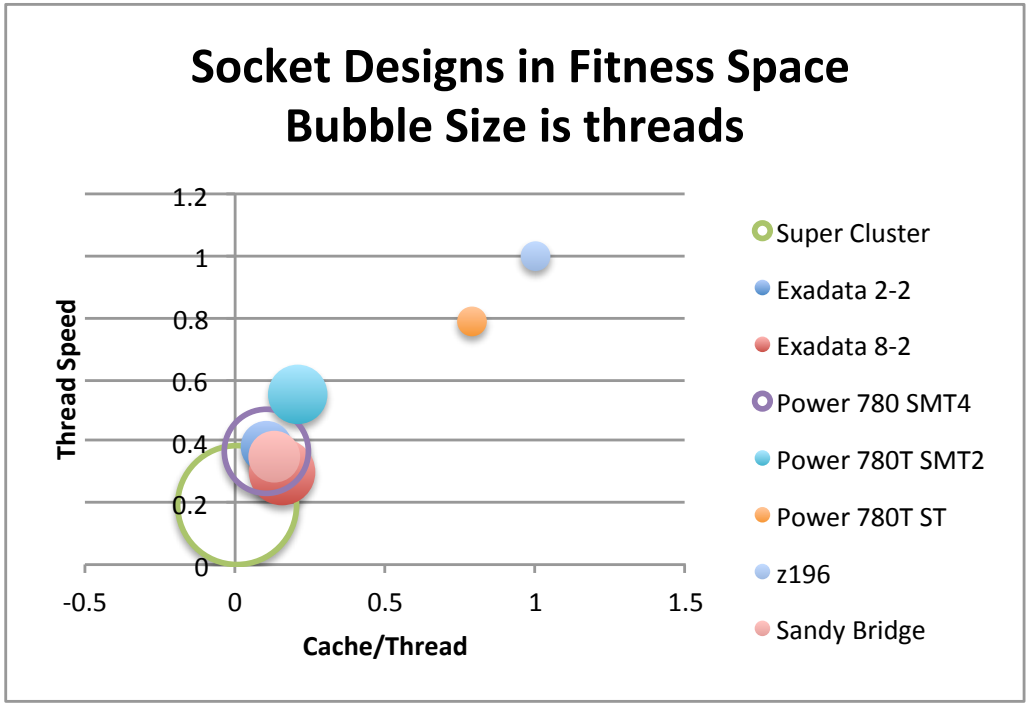
… you don’t know what you think you know.

Sooner or later you will be misled.

Therefore an administratively established Standard Metric will lead to losses.

The Reality

- Relative Capacity dependent on at least three machine parameters:
  - Thread Speed
  - Thread Count
  - Cache/Thread

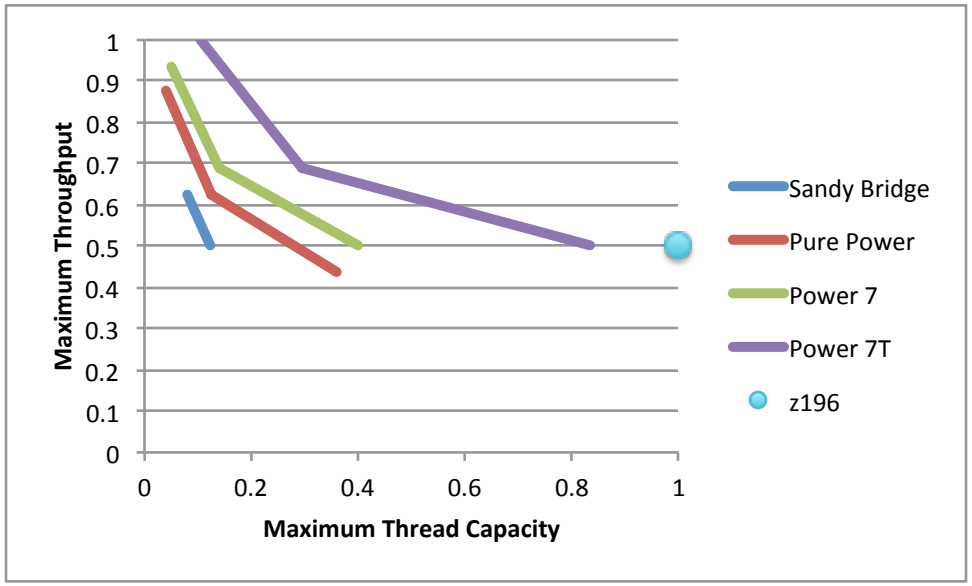


Note that high thread count means reduced thread speed and cache per thread

A Model

We observe:

- Maximum achievable Throughput is Thread Speed x Thread Count
- Maximum Thread Capacity is Thread Speed x Cache per Thread



There is a clear trade off between Throughput and Thread Capacity
Enterprise servers are distinguished by higher thread capacity.
Pure Systems compute nodes generally have low thread capacity but can have high throughput.
Pure Systems aggregate more nodes in a rack

Capacity

- We assert:
  - Total Capacity is a combination of Throughput and Thread Capacity
  - Relative Capacity varies with workload because the relative weight of each parameter is workload dependent.
- We define:
  - Capacity = w(Throughput) + (1-w)Thread Capacity
  - C = w(TC) + (1-w)TP

Relative Capacity

- Relative Capacity is Capacity Ratio between Machines A and B often expressed as a core ratio
  - Relcap =CapA/CapB = Cores B/Cores A
  - Model:

  Relcap = (w(TCA) + (1-w)TPA) / (w(TCB) + (1-w)TPB)

- The following work normalizes everything to the Intel Sandy Bridge Core

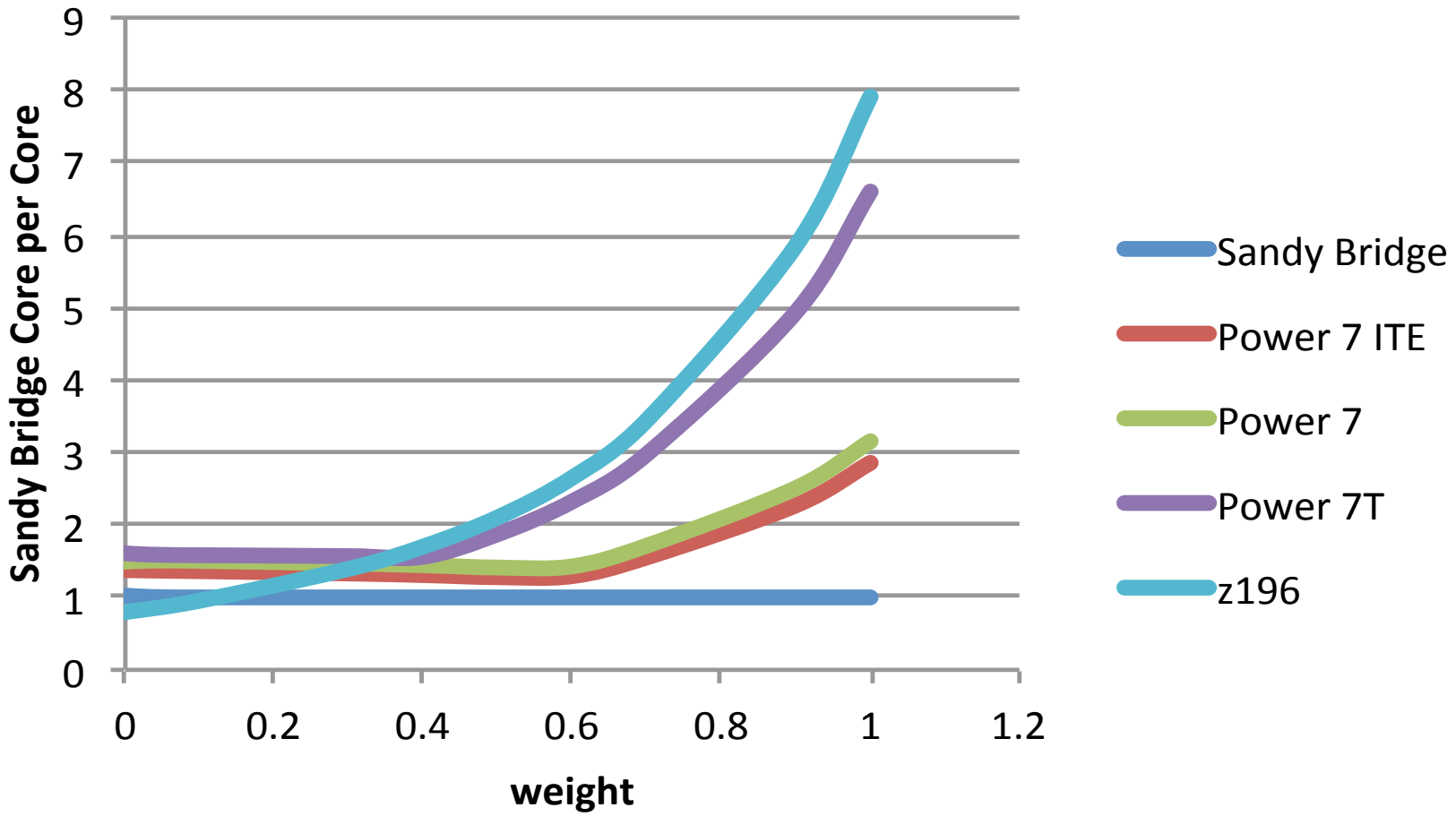
Workload Weight and Relcap

- By our definition weight increases as a workload exploits thread capacity
  - Increased cache misses
  - Increased dependence on a single thread
- Cache misses increase with data intensity of individual loads and with VM or application density when consolidating loads
- Serial Dependence increases with data and resource sharing. This happens by definition in consolidations

Core ratios and weight

- Over the years we have published various descriptions of workload factors and “core ratios” comparing z to other machines.
- The paper “Relative Capacity and Fit for Purpose Platform Selection” (CMG #123, March, 2009) contains a set based on various benchmarks and “real world” comparisons
- IBM has done internal benchmarking (Friendly Bank, single threaded cpu measurements, etc.)
- The next 3 charts are a composite based on all that work.

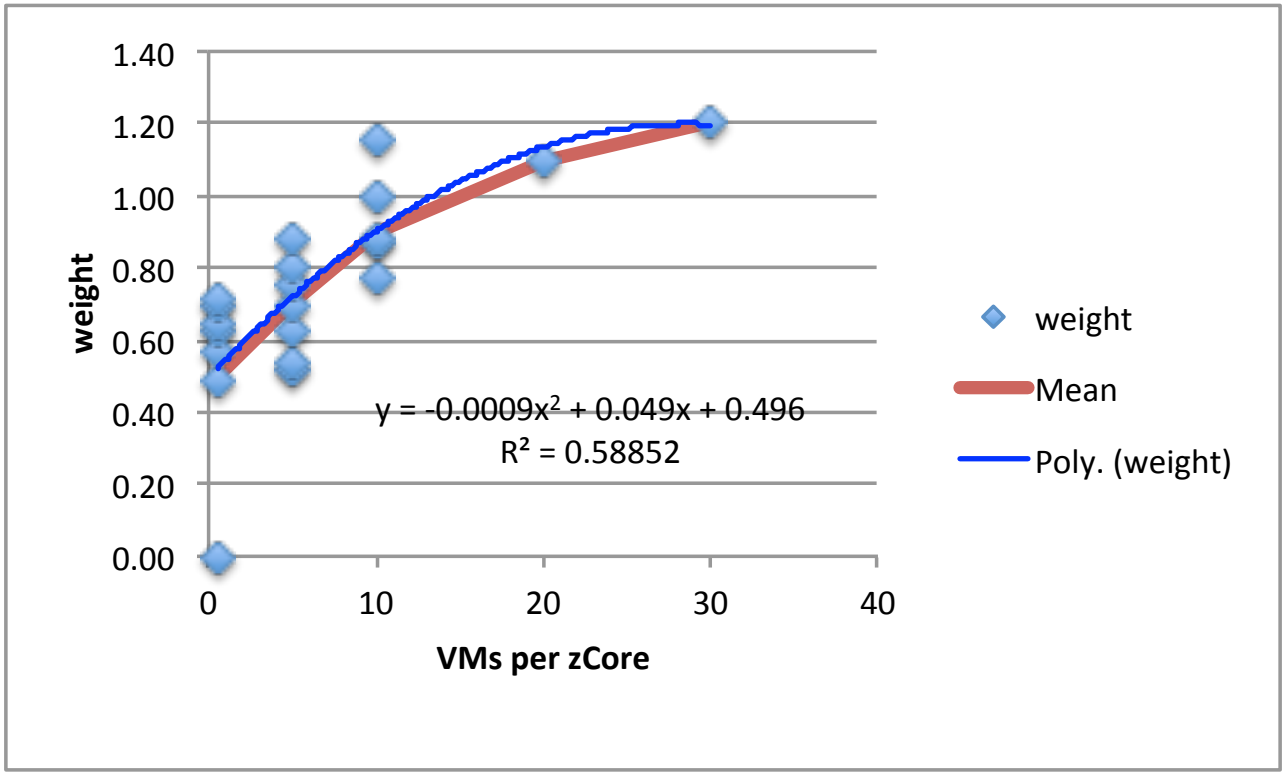
Core Ratios



Workloads and Weight

	VMs/Core	weight	Mean
SAPs	0.5	-0.13	0.5
Throughput Benchmarks	0.5	0.00	0.5
Threaded Integer Processing	0.5	0.49	0.5
Partitioned OLTP	0.5	0.57	0.5
Java OLTP	0.5	0.62	0.5
Web Transactions	0.5	0.65	0.5
Single threaded Integer processing	0.5	0.70	0.5
Java Heavy	0.5	0.72	0.5
TPoX 5VMs	5	0.52	0.7
TPoX 10 VMs	5	0.53	0.7
TPoX 20VMs	5	0.62	0.7
Vitrualization Benchmark	5	0.69	0.7
TPoX 40VMs	5	0.75	0.7
TPoX 80 VMs	5	0.81	0.7
Scaled Virtualization Benchmark	5	0.88	0.7
Friendly Bank Heavy I/O	10	1.16	0.9
Friendly Bank VM Light	10	0.77	0.9
Friendly Bank Raw	10	0.87	0.9
Power VM Max Density	10	0.88	0.9
Thread Capacity	10	1.00	0.9
20 VMs/Core	20	1.10	1.1
30 VMs/Core	30	1.20	1.2

VMs per Core and weight



Note: weights > 1 are due to either I/O or Power VM's product limit of 10 VMs per Core
Power probably will raise the limit some if they build and leverage more thread capacity.
Some metrics like SAPs (SAP SD Benchmark) have weights <0 due to software differences.

Load Variability Effects

- All of the above is about steady state load
- Random variability introduces consolidation and queuing effects
- High variability implies low resource utilization
- Consolidation increase utilization by decreasing variability in the composite load
- Consolidated loads are heavier than the individual loads from which they are built.
- Consolidation leads to more emphasis on thread capacity

Local Factors Govern Variability and Client Response to it

Variability is driven by a combination of market dynamics and business process variation

In the face of variability clients must make operational tradeoffs between throughput, response time and utilization efficiency

Absent batching and buffering, variability effects are stronger on distributed solutions than on centralized solutions.

The operational tradeoffs are governed by Normalized Headroom (HR)

- HR = Capacity Remaining/Capacity Used
  - HR = (1-u)/u
  - As u → 1, HR → 0. As u → 0, HR becomes unbounded
  - Rearranging: Uavg = 1/(1+HR)
- We also know that
  - Uavg = 1/(1+kc/SQRT(n)) Rogers' Equation
    - k is the number standard deviations of the load from the mean at peak utilization design point.
    - c is the standard deviation / mean of a single load instance
    - n is the number of loads on the server.
      - *Note that the n can be less than 1 when a load is distributed*
- Therefore at average utilization:
  - HR = kc/SQRT(n)

Analysis of HR

- Given HR = kc/SQRT(n) we can see:
  - Service Level requirement impact
    - As k → 0, HR → 0 and Uavg → 1
    - As k gets large, HR gets large and Uavg → 0
    - At k = 1, HR is governed by c and n
  - Load Variability impact
    - As c → 0, HR → 0 and Uavg → 1
    - As c gets large, HR gets larger and Uavg → 0
    - At c = 1, HR is governed by k and n
  - Scaling Impact (Variability as a function of n)
    - As n→ 0 (Distribution of load) , HR gets larger and Uavg →0
    - As n gets larger, HR → 0 and Uavg→1
    - At n = 1 HR is governed by k and c

Note that as n increases so does the "weight" of the load.

Response time

- (We will assume here that response time is measured on a single thread of work. When this is not the case, the base response time can be adjusted)
- t = F(1/Capacity, Variability, utilization)
- t_0 ~ 1/Capacity (Link to the raw capacity model)
- Variability = c/SQRT(n) (Link to Rogers' equation)
- t =t_0 + t_{wait} = t_0 + (t_0)(c^2/n)(u/(1-u))
- t =t_0 + t_{wait} = t_0 + (t_0)(c^2/n)/HR) (By definition of HR)
  - t(Uavg) = t_0 + (t_0)(c^2/n)/(kc/SQRT(n) = t_0 + (t_0)(c/kSQRT(n))

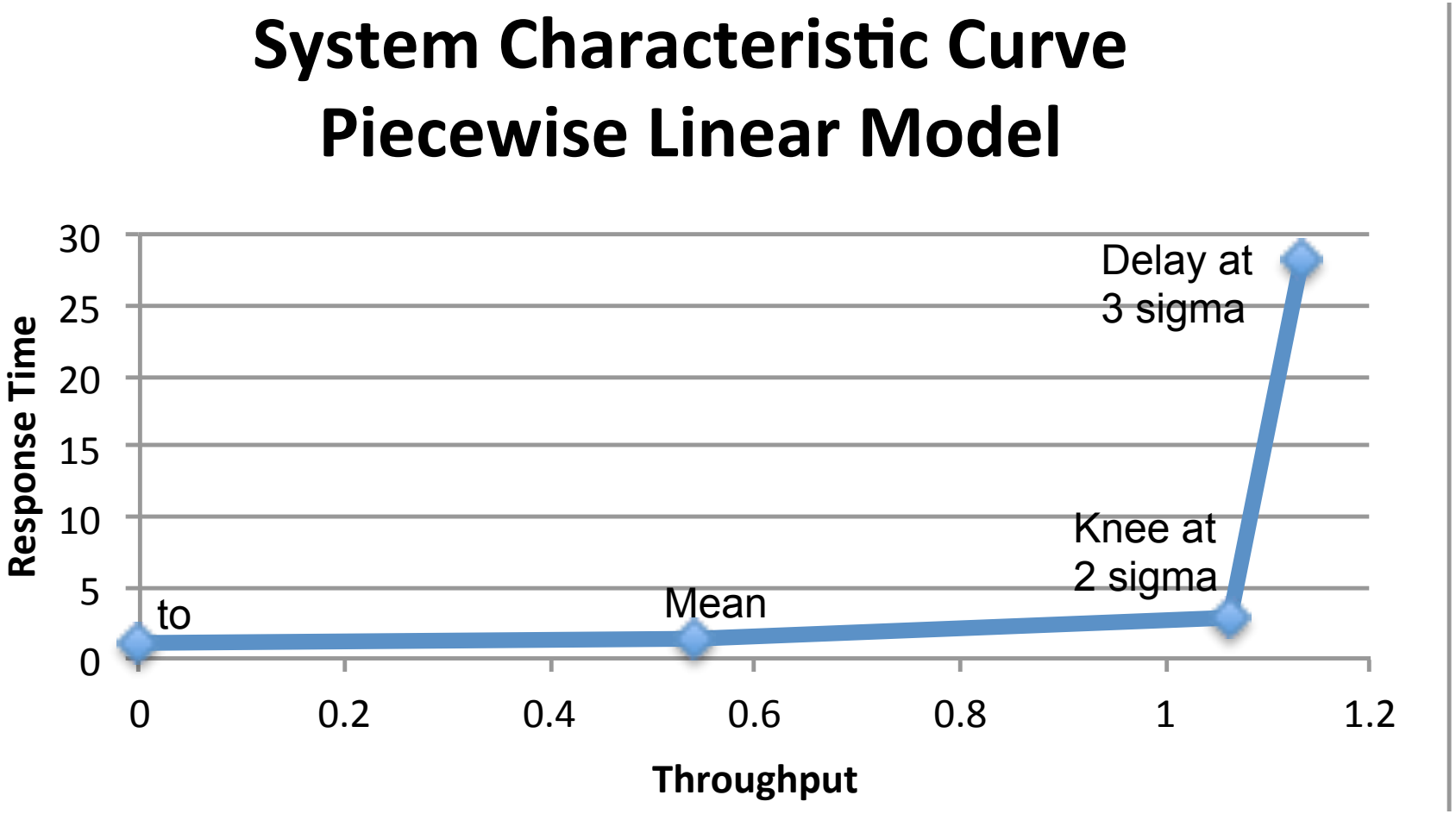
Analysis of Peak Response Time

- Given
  - $t = t_0 + t_{wait} = t_0 + (t_0)(c^2/n)/HR)$
- As capacity goes up t goes down with t_0
- As variability increases the wait time increase with c^2 taking t up with it.
- Consolidation decreases variability, distribution increases it.
- As the maximum utilization increases HR decreases and t goes up

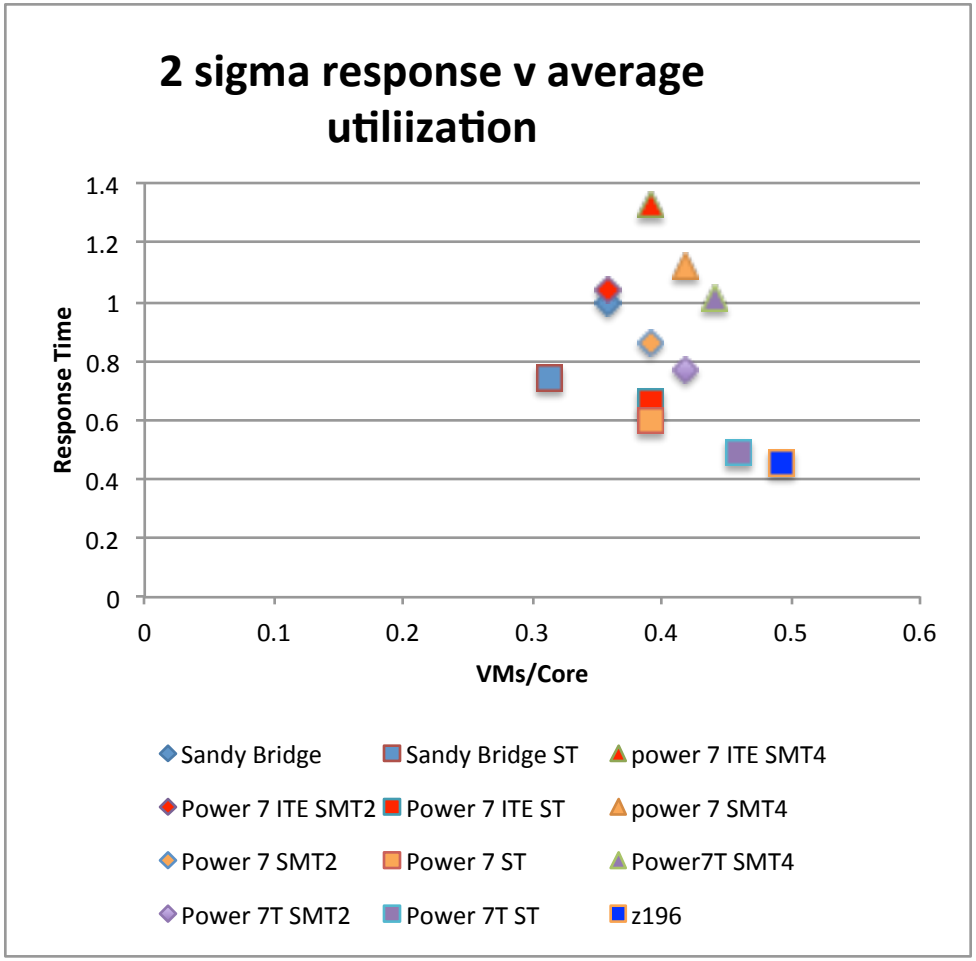
Analysis of Average Response Time

- Given:
  - t(Uavg) = t_0 + (t_0)(c/kSQRT(n))
- As capacity goes up t goes down with t_0
- As variability increases the wait time increase with c taking t up with it.
- Consolidation decreases variability, distribution increases it.
- As SLA gets tighter k is increased reducing the average response time

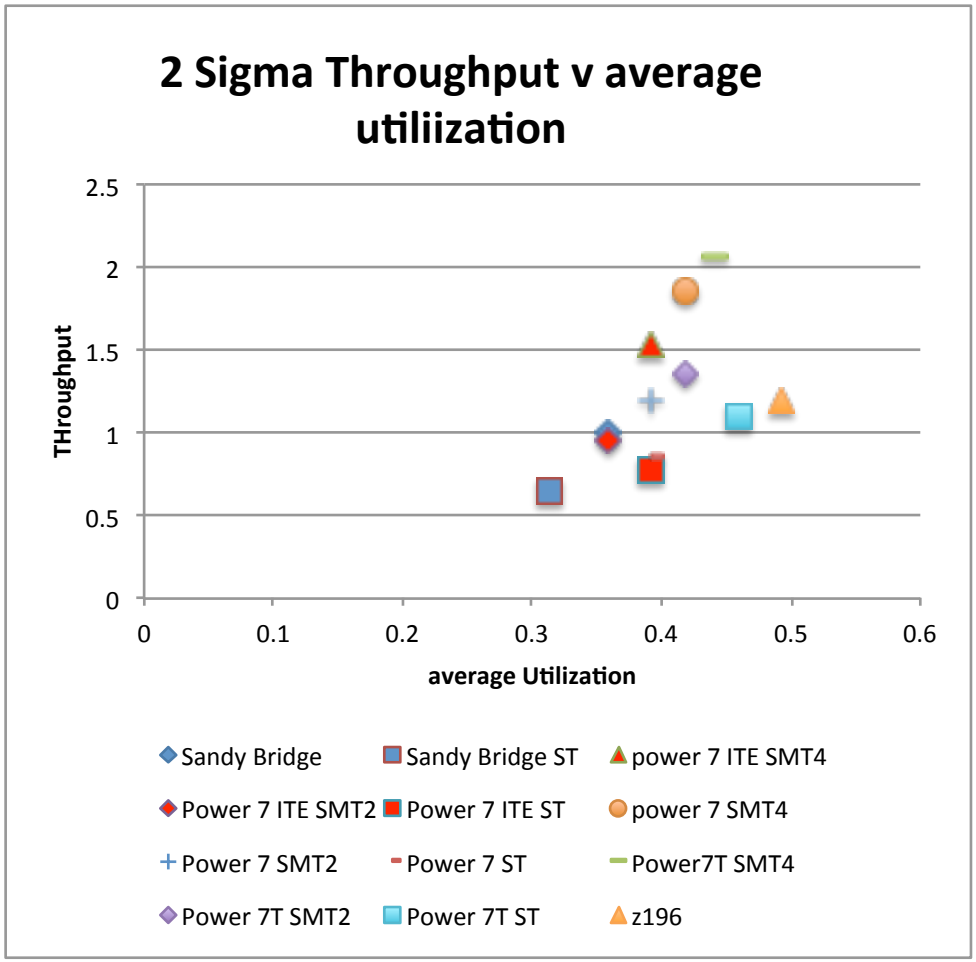
Assume design for good service level at 2 sigma



Trading off efficiency and service level

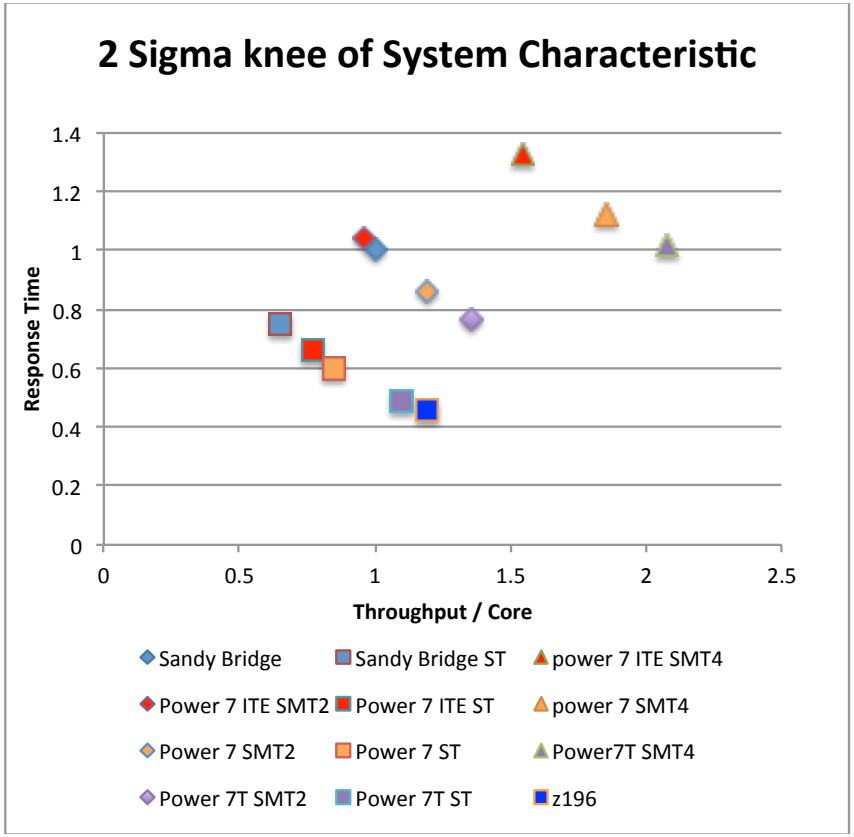


Trading off efficiency and Throughput

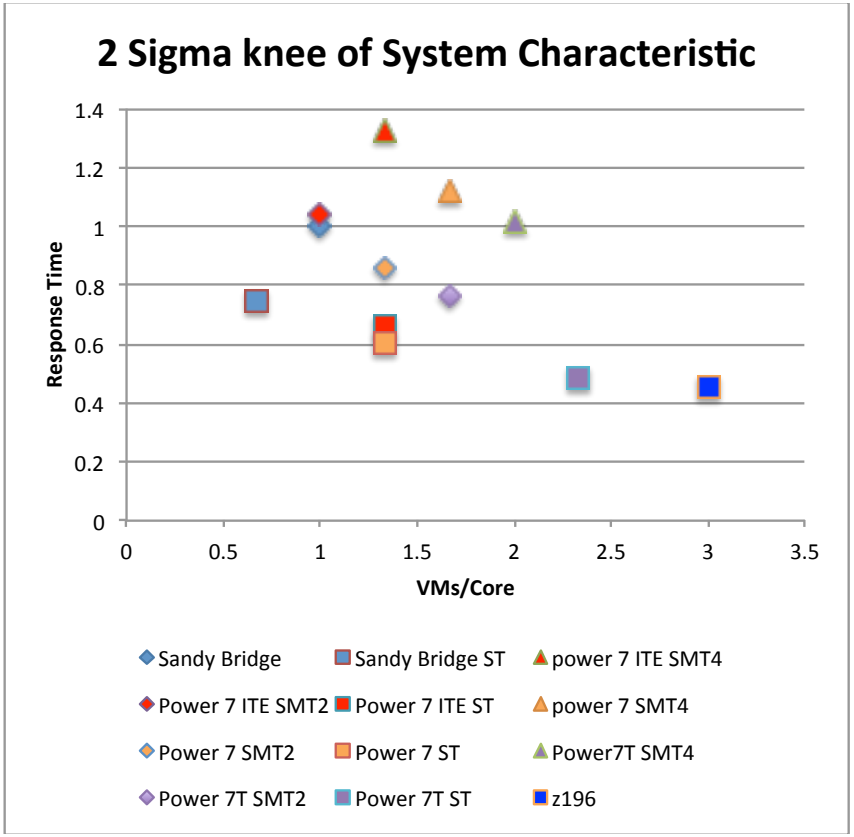


Throughput and Capacity are not the same thing

Throughput/Core



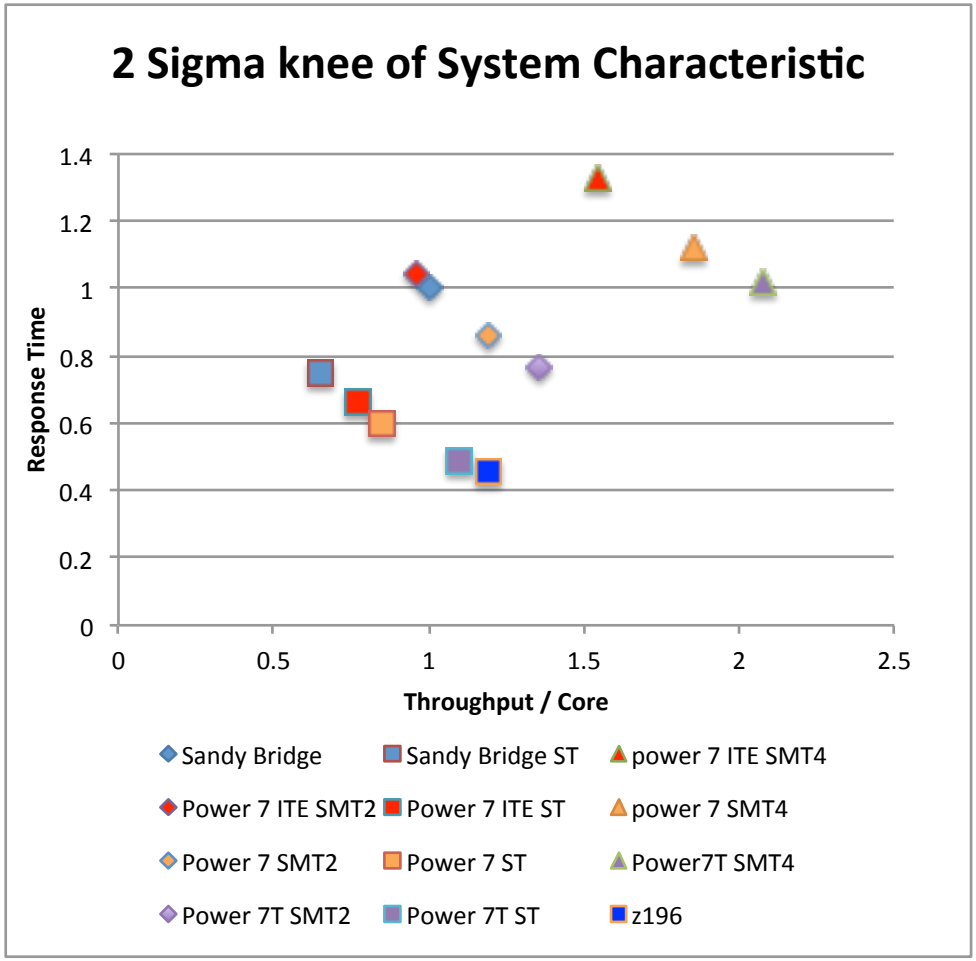
VMs/Core



Moderate weight and variability

k(100%)	3.1
c	1
w	0.5
N Sandy Bridge	3

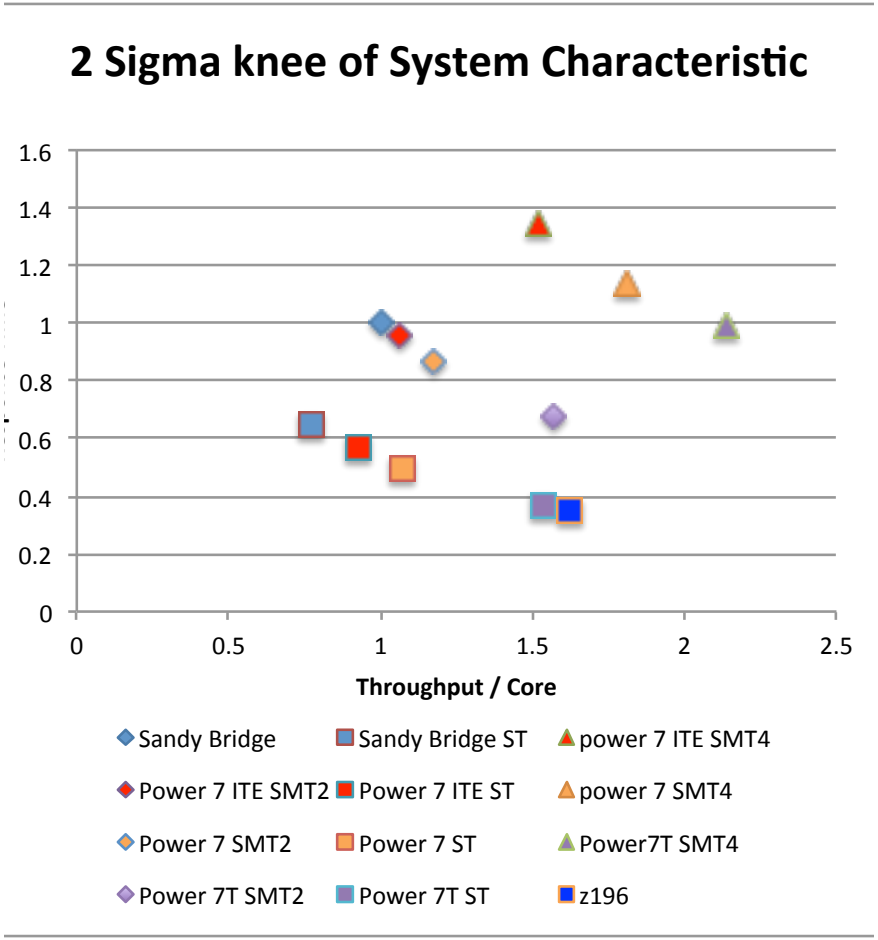
- Variability is Moderate
- Weight is Moderate
- 3 VMs per Sandy Bridge Core
- 5 VMs per Power 7 Core
- 9 VMs per z196 Core
- Response time is single threaded



Consolidation of highly variable loads

k(100%)	3.1
c	2
w	0.7
N Sandy Bridge	4
Thread Unit	1

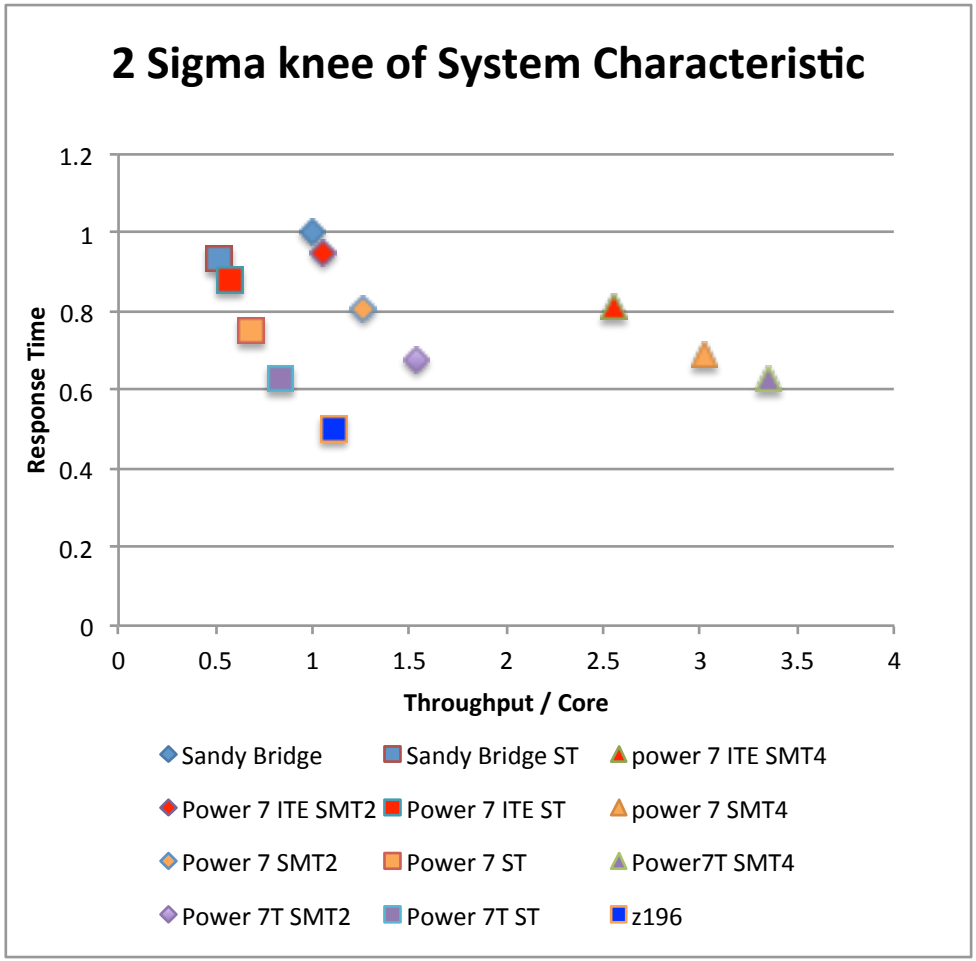
- Weight is high
- Variability is high
- 4 VMs per SB core
- 9 VMs per Power 7 Core
- 25 VMs per z196 Core
- Response time is single threaded



Coarse Grain N <1 for Sandy Bridge

k(100%)	3.1
c	1
w	0.5
N Sandy Bridge	0.5
Thread Unit	4

- Variability is Moderate
- Weight is Moderate
- .5 VMs per SB Core
- 1 VM per Power 7 Core
- 2 VMs per z196 Core
- Response time is multithreaded



Low Variability and Low Weight

k(100%)	3.1
c	0.5
w	0.1
N Sandy Bridge	0.5
Thread Unit	4

- Variability Low
- Weight Low
- .5 VMs per SB Core
- .92 VMs per Power 7 Core
- .47 VMs per z196 Core
- Response Time Multithreaded