Cloud Computing, Data Handling and HPEC Environments

Patrick Dreher
RENCI Chief Scientist Cloud Computing
NC State University Adjunct Professor of Computer Science

RESEARCH \ ENGAGEMENT \ INNOVATION

Outline

- Introduction
- Overview basic cloud computing characteristics
- Some characteristics for HPEC environments
- Large volumes of data – what does that mean?
- Single core versus multi-socket, multi core processors
- Closing comments and observations

Introduction

- Many systems today consist of both remote sensors/instruments and central site capabilities
- Remote sensors/instruments are capable of generating multi-terabyte data sets in a 24 hour period
- Custom clouds architectures appear to have great flexibility and scalability to process this data in an HPEC environment
- How and where should be data be stored analyzed and archived
- How best can this data be transformed into useful actionable information

Basic Cloud Computing Characteristics

Basic Cloud Computer System Design Goals

- Reliable, secure, and fault-tolerant
- Data and process aware services
- Secure dropped-session recovery
- More efficient delivery to remote users
- Cost-effective to operate and maintain
- Ability for users to request specific HW platforms to build, save, modify, run virtual computing environments and applications that are:
  - Reusable
  - Sustainable
  - Scalable
  - Customizable
- Root privileges *(as required/authorized)*
- Time and place independent access
- Full functionality via consumer devices and platforms

A List of Cloud Computing Characteristics

- An operational paradigm allowing the **users** to **seamlessly and securely provision and/or combine**
  - Computer hardware
  - Operating systems
  - Application software
  - Storage
  - Rich set of customizable services
- As a system that is **scalable up, down, in, and out**
  - With resources accessible over a network
  - Based on a service-oriented architecture
- With **users controlling the options** to reserve each equipment and service capability on a **mix-n-match component or unit basis**
- **Implication -** Capability for on-demand provisioning provides the appearance of infinite computing resources available on demand to rapidly follow and adjust to load surges

Clouds Grouped by Services

- **Hardware as a Service (HaaS)** – On demand access to a specific equipment configuration possibly at a particular site
- **Infrastructure as a service (IaaS)** – On demand access to user specified hardware capabilities, performance and services which may run on a variety on hardware products
- **Platform as a Service (PaaS)** - On-demand access to user specified combination of hypervisors, operating systems and middleware that enables applications and services
- **Application as a Service (AaaS)** - On-demand access to user specified application(s)
- **Software as a Service (SaaS)** - may encompass anything from PaaS through AaaS
- **Cloud as a Service –** On demand ability to construct a local cloud within an overall cloud service
- **Security as a Service** – On-demand use of cloud configuration for security of applications and systems

HPEC Characteristics

HPEC Design Parameters

- Usually designed to perform a limited number of dedicated functions often with real-time computing constraints
- Embedded systems mostly integrated within a larger device including hardware and mechanical parts that serve a more general purpose
- The program instructions
  - stored in read-only memory or flash memory chips
  - run with limited computer hardware resources: little memory and small or non-existent keyboard and/or screens

Tradeoffs: Specificity versus Flexibility

- General purpose computational systems capable of multiple types of tasks
- HPEC system configured more toward single/limited type of functionality
- Example of a system configured to do one type of function very well
- Supercomputer capability using mobile phone hardware
- Linpack benchmarks installed on mobile phones running Android OS

  http://www.walkingrandomly.com/?cat=35
- The benchmarks demonstrated that a tweaked Motorola Droid is capable of scoring 52 Mflops i.e. 15X faster than the 1979 Cray 1 CPU
- This capability can be embedded in a larger overall system
- Appears very impressive but ….
- Many real world problems and applications today are driven by data …..lots of data

Observations of Data Characteristics

- In both defense systems and academic research areas
  - There are sensors and experimental laboratory equipment today that can generate multi-terabyte data sets per day that need to be analyzed
  - There are sophisticated SAR image processing and other persistent surveillance platforms capable of producing daily multi-terabyte data sets
- Data processing pipelines need to be constructed to transform this data into useable actionable information
- How can this compute components integrate with the data repositories?

“Large” Volumes of Data

Basics Design Questions For Data Handling with Remote/Field and Central Site Environments

- What does it mean to have a “large volume” of data?
- Where is the data collected? (field, instruments, remote sensors, etc.)
- Where should data be stored?
- Where should the data be analyzed?
- Where should the data be archived?
- Can these multiple environments be integrated?
- Under what conditions
  - Should a customized cloud cluster be constructed for analysis
  - If so, where (HPEC site or central environment)
  - Is it more effective for the data at remote sensors and instruments to be transmitted to a central cloud site or analyzed remotely?

Back of the Envelope Calculation – Analyzing Data Processing Pipeline Options

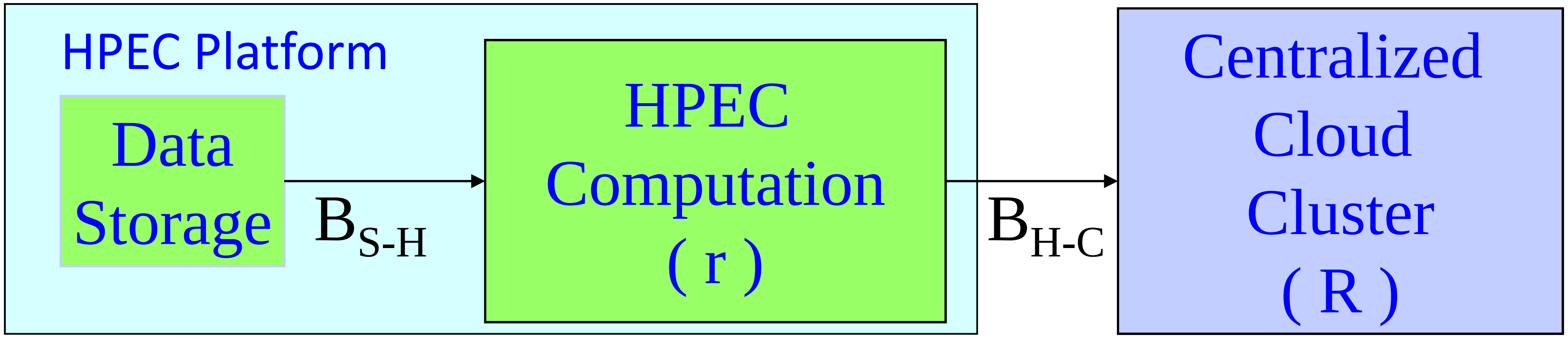
- Using a simple naive toy model - compare the time to process at HPEC site versus the centralized computation site
- What conditions indicate remote or central site as the preferred analysis location
- <u>Procedure</u> -- Reduce the volume of data from V bytes to v bytes and determine whether computation should be done at HPEC site or a central cloud cluster
- Relationship between
  - Execution rate (R)
  - Data access bandwidth (B)
  - Computational complexity (operations per byte) ($\eta = R / B$)
- For a balanced application on a given networked hardware platform

  Complexity = Execution Rate / Bandwidth = 1
- Data is processed at HPEC location corresponds to low complexity
- Data is processed at central location corresponds to high complexity

Design “Thought Experiment”

- Parameters

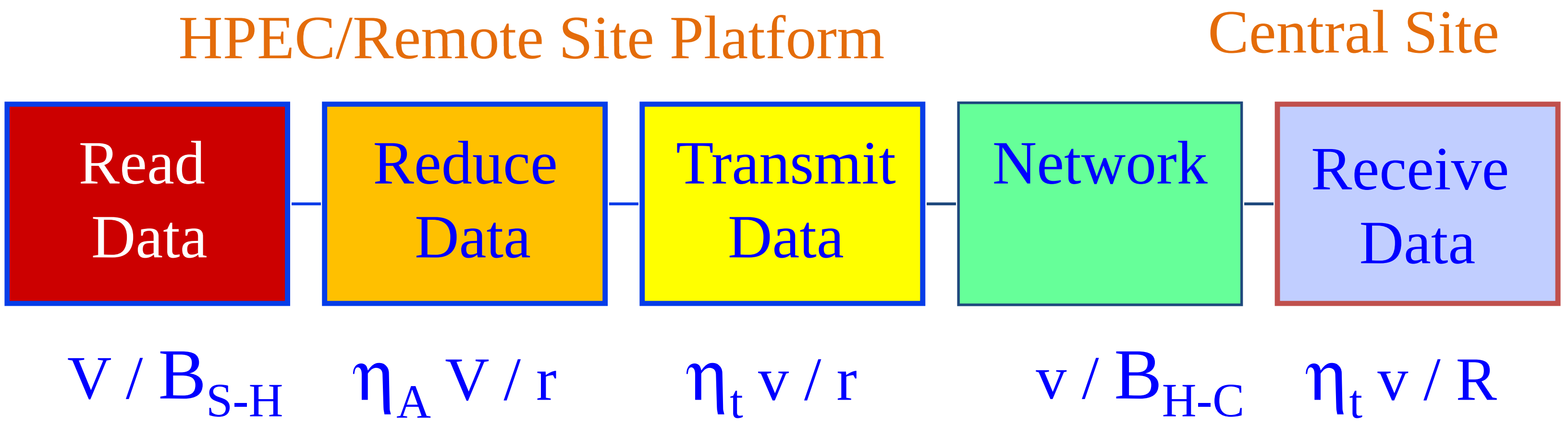
– Execution rate at HPEC site	r
– Execution rate at central site	R
– Bandwidth at HPEC proc - storage	$B_{S\text{-}H}$
– Bandwidth HPEC – central site	$B_{H\text{-}C}$
– Ops/byte analysis at HPEC site	η_A
– Ops/byte for transfer to central site	η_t



Two Choices

- OPTION 1 - Cloud calculation and data reduction at HPEC site and then transmit reduced data centralized cloud location
- Time for each discrete step

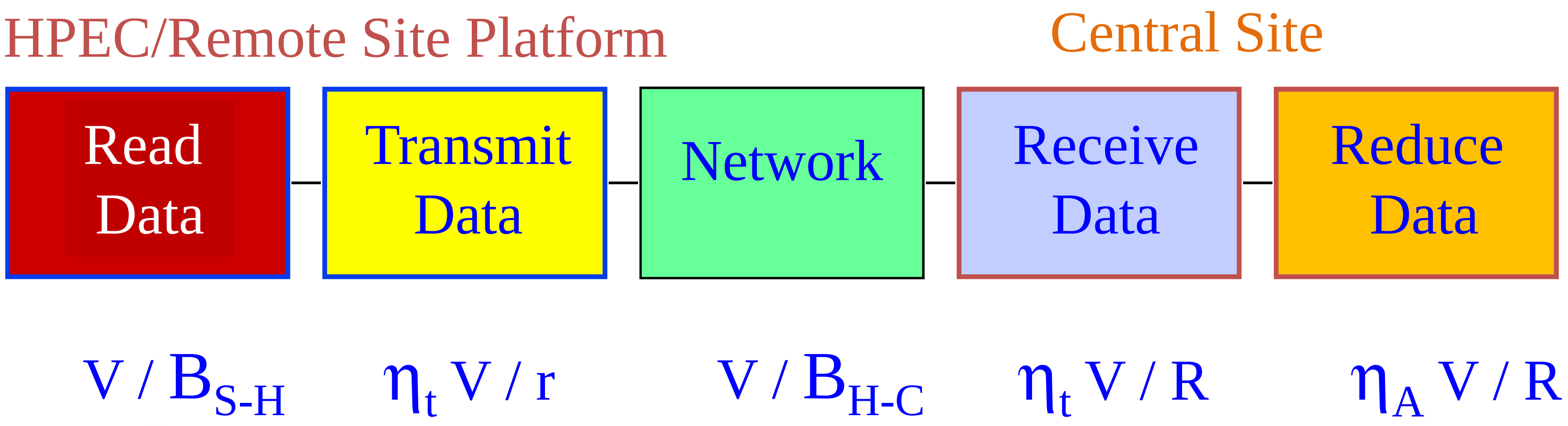
– Reduce the data	$\eta_A V / r$
– Transmit the reduced data	$\eta_t v / r$
– Network transmission	$v / B_{H\text{-}C}$
– Receive data	$\eta_t v / R$



Two Choices

- OPTION 2 -Transmit data from HPEC site and then calculate and reduce data at centralized cloud location
- Time for each discrete step

– Transmit the original data	$\eta_t V/ r$
– Network transmission	$V/ B_{H\text{-}C}$
– Receive data	$\eta_t V/ R$
– Reduce the data	$\eta_A V/ R$



Where to Analyze the Data

- The data should be processed at the remote HPEC site when the total time at remote HPEC site < total time at central site

$$V/B_{S\text{-}H} + \eta_A (V/r) + V / B_{H\text{-}C} + \eta_t (v/r) + \eta_t\, v / R \;<$$
$$V / B_{S\text{-}H} + \eta_t\, V / r + V / B_{H\text{-}C} + \eta_t\, V / R + \eta_A\, V / R$$

- Define several scaling ratios

– Data size reduction ratio	v / V
– Execution slow down ratio	r / R
– Problem complexity	η_t / η_A
– number of bytes/sec that can be processed	(r / η_t)
– Execution/Communication *	$r / (\eta_t B_{H\text{-}C})$

* Optimal design $r/(\eta_t B_{H\text{-}C}) = 1$

Conditions

$$B_{H-C} > \frac{1}{-\frac{\eta_t}{\eta_A}\left(\frac{1}{R}+\frac{1}{r}\right)+\left(1-\frac{v}{V}\right)\left(\frac{1}{r}-\frac{1}{R}\right)}$$

- Examine when the denominator changes sign

$$\left(\frac{1}{1-\frac{v}{V}}\right)\left(\frac{R-r}{R+r}\right) < \frac{\eta_t}{\eta_A}$$

- Denominator > 0 implies better to move the data to central site
- Always do computation at HPEC site if $\frac{\eta_t}{\eta_A} < 1$ and design flexible scalable HPEC multi-socket, multi-core computing resources available on demand to rapidly follow/adjust to load surges
- This cannot be fully accurate because the actual HW systems are more complex and require some additional perspectives

Traditional Memory Performance Metrics in Computational Systems

- Processing large data sets requires ability to rapidly analyze this stored information
- Most characterization methods for memory performance use two measures, memory latency and bandwidth *
  - Memory read *latency is the time between from issuance of a memory* request and the moment when the data is available in the CPU
  - Memory *bandwidth is the rate, usually expressed in bytes per* second, at which a sequence of reads, writes, or some mix can transfer data between memory and the CPU.
- Both measures depend on a set of complex factors and vary greatly depending on offered load within a system

* System memory performance characterizations usually measured with LMBench and STREAM benchmarks

Multi-Socket-Multi-Core Systems

- Multi-socket, multi-core (MSMC) blades and servers now dominate the options for cloud computation systems
- These systems exhibit complicated bottlenecks at several levels
- MSMC Characteristics
  - Multi-core processors exploit parallelism with multiple threads
  - Faster processors now available but memory cell speed has been nearly constant
  - Multi-core systems are becoming increasingly memory-bound
  - Increasing # of cores -> decreasing memory bandwidth / core
  - Increasing core count -> large # of concurrent memory accesses from multi-threaded applications
- **QUESTION: Can the MSMC computational HW improvements and rapidly expanding data sets be a problem for proposed computationally intensive custom cloud HPEC solution?**

Memory Performance in Multi-Socket Multi-Core Systems

- Bandwidth and latency alone are not sufficient to characterize memory performance of multi-socket, multi-core systems
- Ability of system to serve large number of concurrent memory references is becoming a critical performance problem
- The assertion is that for multi-socket, multi-core systems concurrency among memory operations and ability of system to handle that concurrency are fundamental determinants of performance
- Concurrency depends on several factors
  - how many outstanding cache misses each core can tolerate
  - the number of memory controllers
  - the number of concurrent operations supported by each controller
  - memory communication channel design
  - the number and design of each of the memory components

Recent Studies on Concurrency in Multi-Socket Multi-Core Systems

- There has been some in-depth analysis of issues of concurrency in newer multi-socket, multi-core computational systems at RENCI*
- In this work, Mandal et. al. treat memory concurrency as a fundamental quantity for modeling memory behavior for multi-socket multi-core systems

* A. Mandal, R. Fowler and A. Porterfield, "Modeling Memory Concurrency for Multi-Socket Multi-Core Systems", in Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'10), pp. 66-75, White Plains, NY, March 2010.

Observed Impacts of Concurrency on MSMC

- In the multi-socket, multi-core domain, simple linear relationships don’t hold true because these systems have non-linear performance bottlenecks at several levels.
- There are multiple points at which memory requests saturate different levels of the memory - bandwidth no longer increases with offered load.
- Memory optimization can be done by increasing the concurrency of memory references rather than just by reducing the number of operations.
- On multi-threaded systems, there are operation points in which performance can increase by adding threads or by increasing memory concurrency per thread, as well as other modes in which increasing program concurrency only exacerbates a system bottleneck.

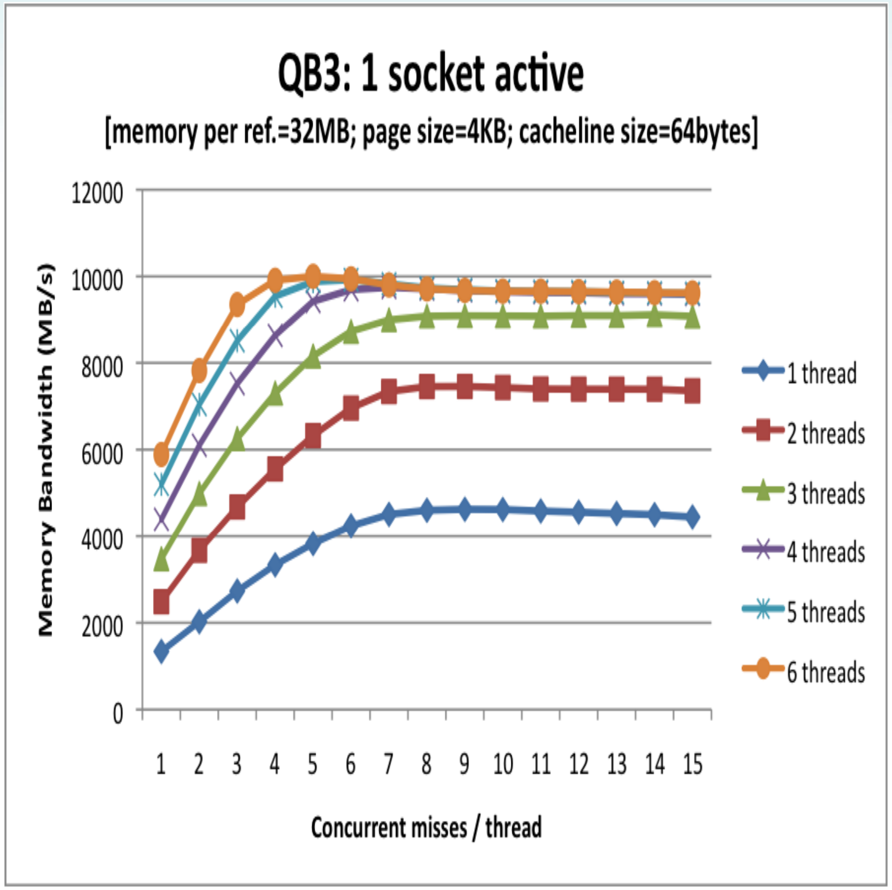
Measuring Concurrency

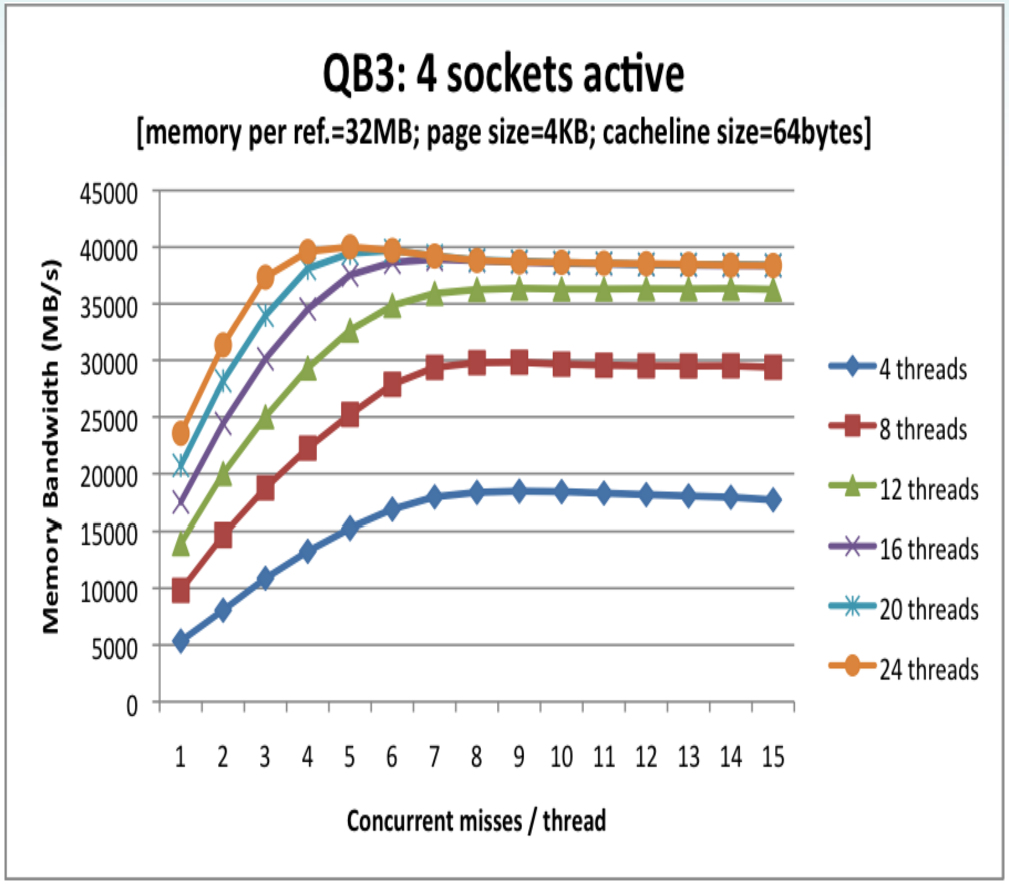
- Use pChase – tool to measure concurrency (developed by Pase and Eckl @IBM and available at http://pchase.org)
- Multi-threaded benchmark used to test memory throughput under carefully controlled degrees of concurrent accesses
- Each experimental run of pChase parameterized by
  - Memory requirement for each reference chain
  - Number of concurrent miss chains per thread
  - Number of threads.
- Page size, cache line size, number of iterations, access pattern kept fixed with pCHASE extended to pin threads to cores and to perform memory initialization after being pinned

pChase Modifications by RENCI

- Each thread executes a controllable number of pseudo random 'pointer-chasing' operations – memory-reference chain (defeats prefetching)
  - pointer to the next memory location is stored in the current location
  - results in concurrent cache misses -> concurrent memory requests
- Modifications by Mandal et. al.
  - Added wrapper scripts around pChase to iterate over different numbers of memory reference chains and threads thereby controlling memory concurrency
  - Added affinity code to control thread placement
- Mandal et. al. tested the procedure on many different systems

A. Mandal, R. Fowler and A. Porterfield, "Modeling Memory Concurrency for Multi-Socket Multi-Core Systems"





QB3 -- Quad-socket, 6-core AMD Istanbul 2.1GHz processors in a Dell PowerEdge M905 (total of 24 cores) running CentOS Linux (2.6.32.7 kernel) with 16 dual-rank 2GB DDR2/800 memory sticks (evenly loaded) for a total of 32GB with each socket having 8G memory available memory

Remarks About the Multi-Socket Multi-Core Tests

- All multi-socket multi-core systems exhibited similar types of behavior
- With low concurrency there would be speedups when increasing the number of threads and/or number of sockets
- With higher increasing levels of concurrency cannot continue increasing peak bandwidth for single socket and multi socket tests
- At some point, the additional cores do not increase the peak bandwidth at all, but they do allow the peak to be reached with fewer offered concurrent memory references per thread
- Reference full paper* for additional system measurements

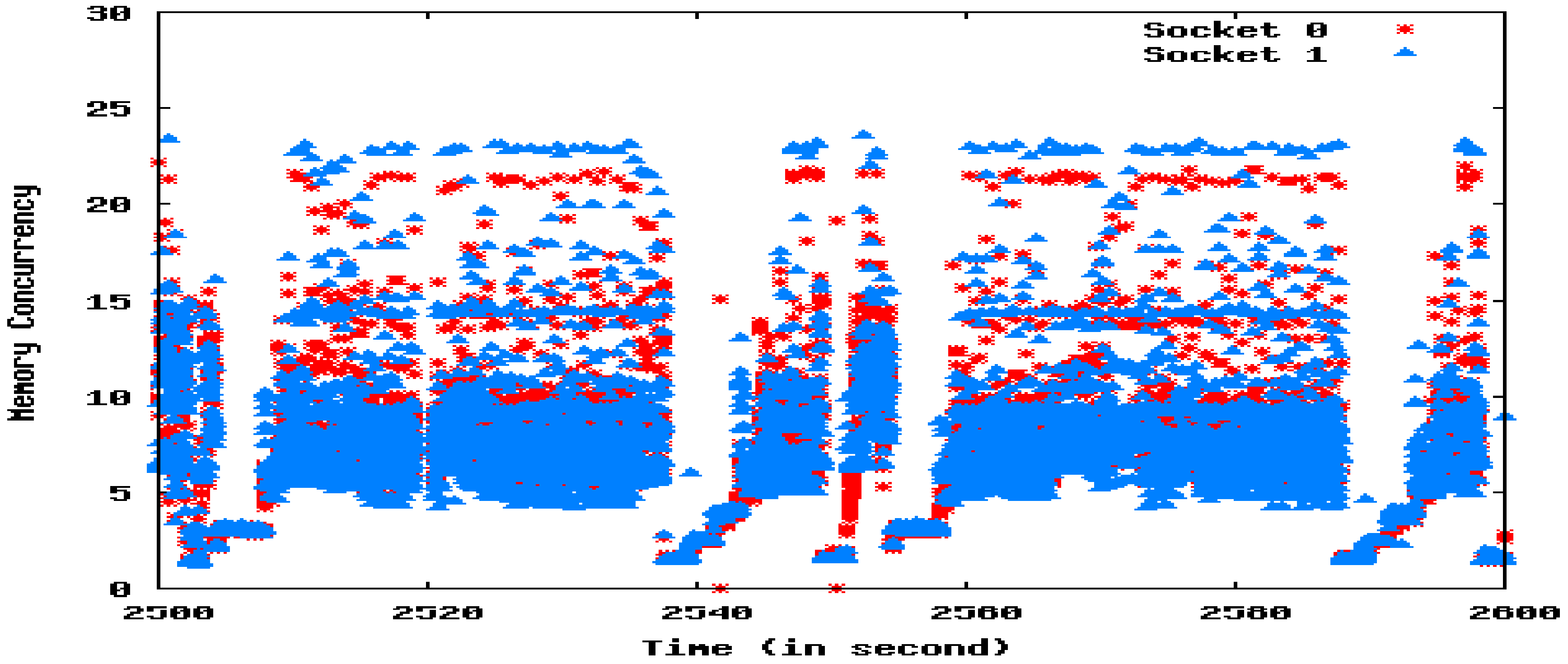
Remarks About GPUs HW Design Versus MSMC Blades and Servers

- The MPPs, SMPs, and commodity clusters rely on *data caches* (small amount of fast memory close to the processor) to contend with the balance issue.
- Modern commodity processors have three levels of cache, with L3 (and sometimes L2) shared among multiple cores.
- Today's GPU hardware architectures are an attempt to re-gain better memory bandwidth performance (ex. The NVIDIA GeForce GTX 285 has eight 64-bit banks, delivering 159 GB/s)
- Compare today's HW architectures to early vector machines that delivered high bandwidth through a very wide memory bus (ex. memory of the Cray X-MP/4 (ca. 1985) was arranged in 32 (64-bit) banks, delivering 128 GB/s)

Comments and Observations

- Memory bandwidth problem
  - Critical path item in many applications for processing large volumes of data
  - Deteriorating with newer generation hardware architectures
- The main bottleneck is often bandwidth to memory, rather than raw floating point throughput
- HW advances since 1985 have trended toward extreme floating point performance (cell phone example) but relatively few improvements in memory bandwidth performance
- For single core -- condition bandwidth, data reduction size and computational complexity impact memory bandwidth
- Today's HW offerings use designs with multi-socket, multi-core systems that exhibit performance bottlenecks at several levels and bandwidth and latency are not sufficient to characterize memory performance
- Use results to develop new tools for tuning domain science codes to HW (for example physics quantum chromodynamics codes)

Performance Measurement for QCD Physics Code on Multi-Socket Multi-Core system *



* Mandal, Lim, Porterfield, Fowler, "Effects of Multi-core Memory Concurrency Limits on Multi-threaded Applications "

Comments and Observations (cont'd)

- Recent experimental measurements at RENCI* indicate the key observation is that the offered concurrency among memory operations and the ability of the system to handle that concurrency are the fundamental determinants of performance
- For multi-socket- multi-core the fundamental quantity for modeling should be concurrency rather than latency and bandwidth
- Designs for HPEC with large volumes of data need to analyze where the data should be computed, with what HW architecture will it be done, and what are the concurrency parameter measurements for that system

* A. Mandal, R. Fowler and A. Porterfield

Discussion and Questions