CS 431/531 Introduction to Performance Measurement, Modeling, and Analysis
Winter 2019

Prof. Karen L. Karavanic
karavan@pdx.edu
web.cecs.pdx.edu/~karavan

Today’s Agenda

- Why Study Performance?
- Why Study Performance Now?
- Introductions
- Course Logistics
- Introduction to Performance Profiling

Parallel Performance Tools (1990s)

1. Goal: Locate the Bottleneck (profiling)

   Applications large/long running

   Processors single core, early clusters, supercomputers

2. Problem: Synchronization (tracing)
   - MPI applications:
     - one MPI process waiting to receive a message, sender is slow
     - Barrier: ALL OTHER PROCESSES waiting for one slow process to reach barrier

3. Scaling
   - Intel Paragon: 1-4000 Processors/Nodes !!
   - How to measure all of the nodes??
   - Perturbation: we can no longer measure everything

Paradyn Parallel Performance Tools

- Key insight (Hollingsworth/Miller 1994): *what if we could insert and remove instrumentation instructions on the fly, as the application runs??*
- Dynamic instrumentation
- Automated Performance Diagnosis
  - Define common problems and “hypotheses”
  - Search through the space of all possible problems (“why”) places in the code (“where”) and phases throughout the long run (“when”)
  - Tool user just pushes a button
- First Demo: SC Exhibit Hall
- My Dissertation: Using historical data to make this more efficient
- Current Work: Autotuning (measure and improve performance as the code runs)

Parallel Performance Tools: 1990s to present

- Threading (Shared address space)
  - Scalability Challenges: 1,000s-10,000 of threads
    - How to measure all of the threads for each node??
    - New challenges: locking, visualization
    - How to visualize??
- Multicore (2007 ->)
  - Scalability Challenges: 2-16 cores/processor
    - How to measure all of the cores for each processor??
      - Outliers vs aggregation
    - Perturbation
    - New challenges: memory, locking

1990s to present (cont'd)

- Manycore
  - Example: General Purpose GPU Computing
    - 1000s of low power compute cores
    - Used as "accelerators" to CPUs
    - New Challenges:
      - High level directives: how to explain problem to programmer?
      - Host <> device communication over PCI bus
- Power/Cooling
  - Great Berkeley Quote: You can't cause more climate problems powering your data center than you solve with your science
  - Exascale Reality: We cannot generate enough power to simply expand our systems
  - New Challenge for tools: measure/report power/heat

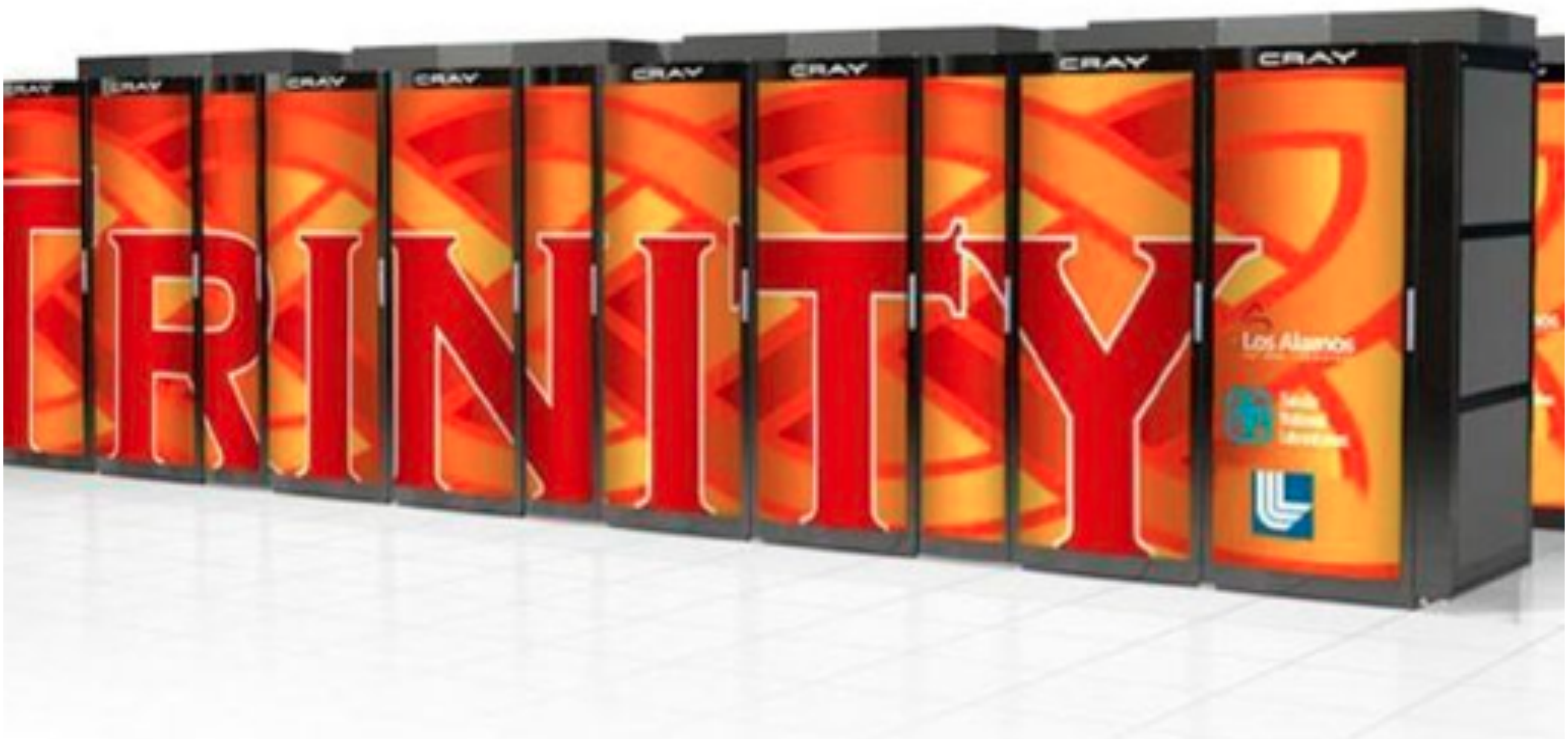
Why Learn Performance? An Example

HPC Today: Trinity @LANL *

- Architecture Cray XC30
- Memory capacity >2 PB of DDR4 DRAM
- Peak performance >40 PFLops
- Number of compute nodes >19,000
- Processor architecture:
  Intel Haswell & Knights Landing ("Xeon Phi")
- Parallel file system capacity (usable) >80 PB
- Parallel file system bandwidth (sustained)1.45 TB/s
- Burst buffer storage capacity (usable) 3.7 PB
- Burst buffer bandwidth (sustained) 3.3 TB/s
- Footprint <5,200 sq ft
- Power requirement <10 MW

* *http://www.lanl.gov/projects/trinity/*

Trinity@LANL



Scaling, Power, Cooling

Scaling Challenges for Performance Measurement (2017)

- We still have perturbation limits…. Only worse
- We still can't collect all of the data into a huge tracefile… only worse
- Now we can't move all of the data through the interconnect or even off of the system
- We have to worry about how much power our tools need ("power cap")
- Simple aggregation may hide performance problems – also need data for outliers
- Drives current performance tool research

Introductions

Introductions: Me

- Hunter College High School (NYC)
  - Intellectually gifted, admission by exam
- Stuyvesant High School (NYC):
  - Math and Science, admission by exam
- New York University: B.A. Computer Science
- University of Wisconsin – Madison:
  - M.S. Computer Science
  - Ph.D. Computer Science
    - WARF Fellow, IBM intern, NASA GSRP Fellow
- Portland State University 2000 ->
  - LLNL, SDSC, New Mexico Consortium
- Current Projects: Holistic HPC Workflow Analysis, SMM-based Runtime Integrity Checking, Undergraduate Curriculum

Introductions: You

- Please use one sheet of paper
  OR one email
  – subject “Performance Intro”
  – To: karavan@pdx.edu
- 1. Your name (include nickname)
- 2. Your degree and progress
  – e.g. M.S. CS expected fall 2017
- 3. Your goals for the course
- 4. Any experience with: parallel programming, Map/Reduce, OS, architecture?
- 5. Performance Topic of particular interest?

Introductions: You <> You

- Please Form groups of 4
  - Introduce yourselves in pairs
  - Now switch (how can you meet everyone??)

- 1. Your name (include nickname)
- 2. Your degree and progress
  - e.g. M.S. CS expected fall 2017
- 3. Your goals for the course
- 4. Your favorite CS course so far (here or elsewhere)

What is Performance Evaluation ?

1. Define the goals and define the system boundaries
2. List system services and possible outcomes
3. Select performance metrics
4. List system and workload parameters
5. Select factors and their values
6. Select evaluation techniques
7. Select the workload
8. Design the experiments
9. Analyze and interpret the data
10. Start over, if necessary

Three Techniques

- Measurement
  - Example: performance study of a new platform
  - Hemmert et al, Trinity: Architecture and Early Experience https://cug.org/proceedings/cug2016_proceedings/includes/files/pap143s2-file1.pdf
  - Vaughan et al, Early Experiences with Trinity - The First Advanced Technology Platform for the ASC Program, Cray Users Group Meeting 2016 https://cug.org/proceedings/cug2016_proceedings/includes/files/pap139s2-file1.pdf
- Modeling
  - Example: Performance Model for an upcoming system
- Simulation
  - Example: Simulate a new Cache Design

Approach 1: Measurement

1. When can we do it?
   Post-prototype
2. How much time is required?
   Varies, usually multiple full application runs
3. What tools do we need?
   HW and/or SW instrumentation
4. How accurate is it?
   Varies with methodology
5. Will it help to evaluate tradeoffs?
   With some difficulty
6. What is the cost?
   Time and resource cost to perform the measurements; data analysis cost
7. Will people believe the conclusions?
   Very well accepted

Approach 2: Modeling

1. When can we do it?
   Any time
2. How much time is required?
   Usually fast; new model development is time consuming for more complex cases
3. What tools do we need?
   “Paper and pencil”
4. How accurate is it?
   Can be low
5. Will it help to evaluate tradeoffs?
   Easy with this method
6. What is the cost?
   Usually low cost
7. Will people believe the conclusions?
   Low

Approach 3: Simulation

1. When can we do it?
   Any time
2. How much time is required?
   Usually moderate; complex simulation is done in parallel and can be very time consuming.
3. What tools do we need?
   Simulation libraries, program development, data storage & analysis
4. How accurate is it?
   Usually fairly accurate, if simulation is correctly designed to represent the actual system
5. Will it help to evaluate tradeoffs?
   Can be used for this
6. What is the cost?
   Moderate (high in the most complex cases eg earthquake simulation)
7. Will people believe the conclusions?
   Medium