# Software Knows Best: A Case for Hardware Transparency and Measurability

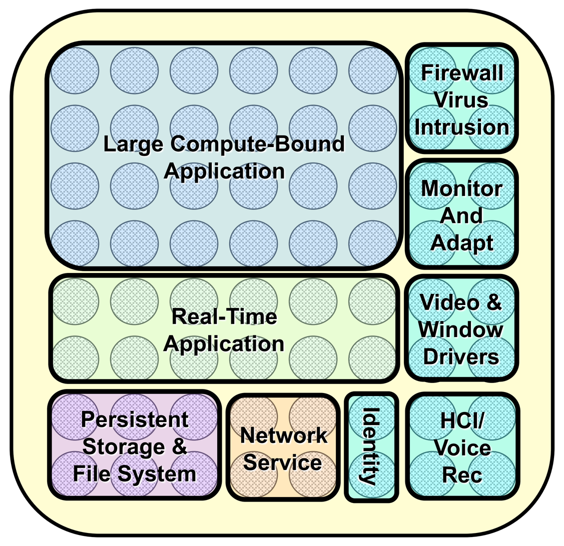
Sarah Bird, Kaushik Datta, Karl Fuerlinger, Archana Ganapathi, Shoaib Kamil, Rajesh Nishtala, David Skinner, Andrew Waterman, Samuel Williams, Krste Asanovic, and David Patterson
slbird@eecs.berkeley.edu

EECS Electrical Engineering and Computer Sciences

BERKELEY PAR LAB

P A R A L L E L   C O M P U T I N G   L A B O R A T O R Y

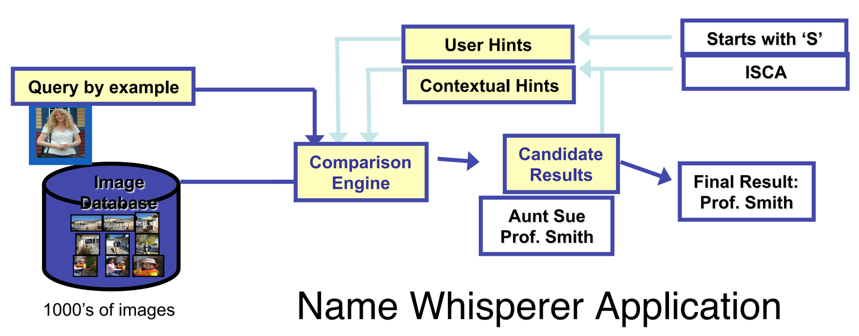
## Manycore Application Development Challenges




- Diverse Platforms
  - "The Laptop/Handheld is the Computer"
  - "The Datacenter is the Computer"
- Split between the Client/Cloud
  - Where to split varies from device to device
  - Offline mode
- Constantly Changing Resource Behavior
  - Other applications running simultaneously
- Efficiency is Important
  - Battery life
- User Driven Deadlines

## Future of Applications

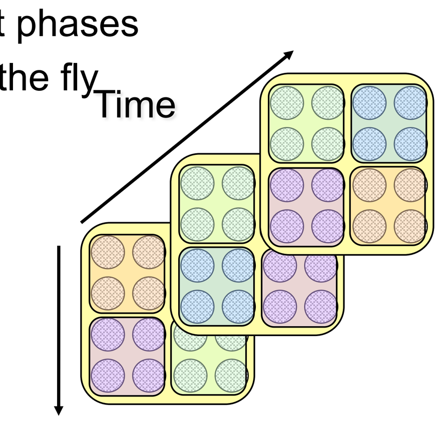
- Complex mobile applications
  - Interactive
    - Responsive
    - Realtime
  - High performance
  - Low battery usage




Name Whisperer Application

## Adaptive Stack

- Operating System
  - Track ALL resource usage of applications in different phases
  - Compute performance-bandwidth-energy curves on the fly
  - Adjust resource allocation for better efficiency
- Applications
  - Schedule threads to avoid contention
  - Adjust to execution environment
    - Select different versions of autotuned code
    - Reduce work to meet deadlines




## Performance Portability Doesn't Exist

7–Point Stencil [Fraction of platform best]

| Platform \ Best Parameter Configuration | Naive | Clovertown | Nehalem | Barcelona | BG/P | Niagara2 |
|---|---|---|---|---|---|---|
| Niagara2 | 0.12 | 0.07 | 0.32 | 0.08 | 0.04 | 1.00 |
| BG/P | 0.23 | 0.63 | 0.75 | 0.83 | 1.00 | 0.51 |
| Barcelona | 0.16 | 0.93 | 0.86 | 1.00 | 0.19 | 0.40 |
| Nehalem | 0.17 | 0.43 | 1.00 | 0.44 | 0.23 | 0.59 |
| Clovertown | 0.52 | 1.00 | 0.94 | 0.95 | 0.42 | 0.78 |

- We tuned a 7-point and a 27-point stencil application for 5 platforms
- We then ran each tuned application and an untuned application on all of the platforms
  - Typical Slowdown was between 1.5x and 3x
  - Code Tuned for Blue Gene always ran slower than untuned code

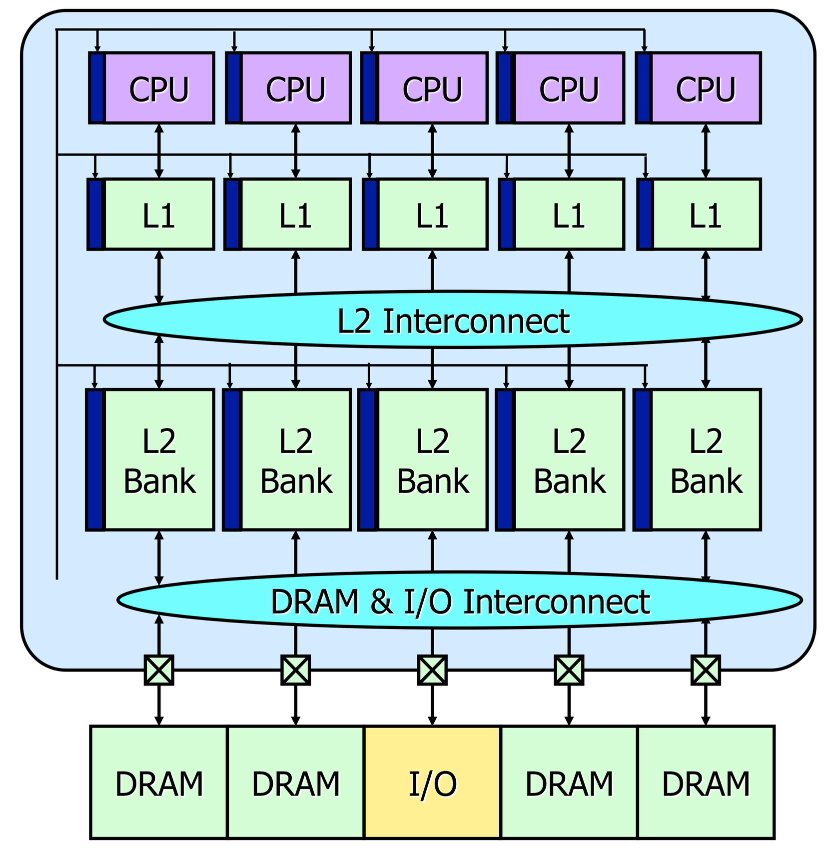
27–Point Stencil [Fraction of platform best]

| Platform \ Best Parameter Configuration | Naive | Clovertown | Nehalem | Barcelona | BG/P | Niagara2 |
|---|---|---|---|---|---|---|
| Niagara2 | 0.55 | 0.05 | 0.23 | 0.05 | 0.04 | 1.00 |
| BG/P | 0.35 | 0.40 | 0.78 | 0.54 | 1.00 | 0.55 |
| Barcelona | 0.25 | 0.75 | 0.74 | 1.00 | 0.24 | 0.65 |
| Nehalem | 0.28 | 0.37 | 1.00 | 0.43 | 0.17 | 0.61 |
| Clovertown | 0.52 | 1.00 | 0.85 | 0.97 | 0.28 | 0.79 |

- On Nigara2 untuned code ran faster than code tuned for any other computer
- Running Blue Gene Code on Nigara2 resulted in a 25x slowdown
- For perf./energy, applications must be tuned for each individual platform
- We can't hand tune every application for every machine so it must be automated

## SHOT

- Create a standard performance measurement system
  - Application level metrics
  - Available on all architectures
  - Consistent access interface
  - Tracks information per task
- Atomically Snapshot Set of Counters
  - Cores have Individual DVFS
    - Use a Global Realtime Clock (GRTC)
    - Much slower than cores
    - Fast enough for apps
    - ~100 MHz
- Apps and OS both need access
  - OS and User Level Latches
- Fast Save and Restore
  - Context switch
  - Hypervisors
- Low Access Overheads




### Energy

- Energy information can affect some non-obvious tradeoffs for applications
  - How much processing to do to compress data before sending it to the cloud?
  - If an app doesn't scale well do we give it more cores?
- Attribute all energy usage to a given component
- Shared resources must split usage by apps

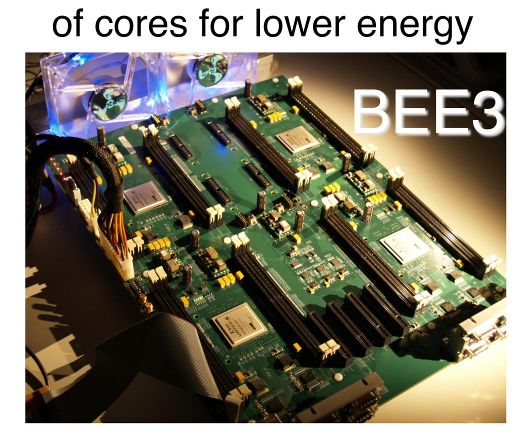
### Comm and Comp

- Communication
  - Interconnect behavior impacts performance
    - Access to DRAM and I/O
    - Communication between cores
  - Measure traffic on every edge
    - Source to Sink
  - Break traffic into causes
- Computation
  - Efficient execution of each core is still important
    - Affects Power/Energy
    - Impacts overall system performance

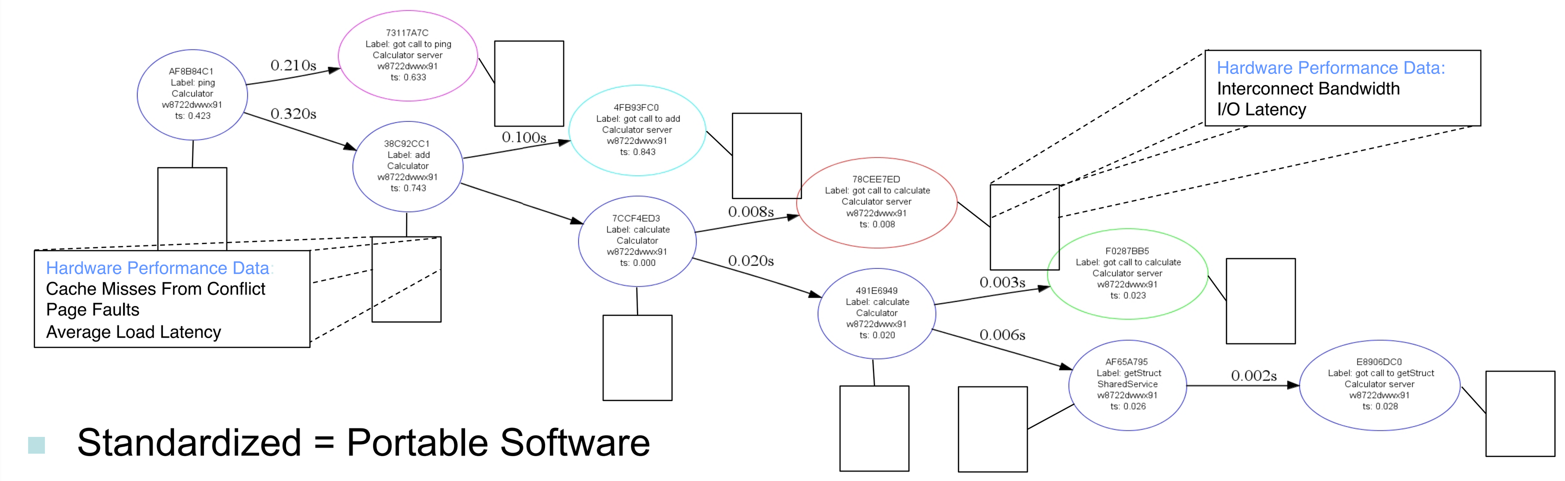
## Our Implementation using RAMP

| Category | Metric |
|---|---|
| Communication | Cache traffic: L1I$, L1D$, L2$, ... |
| | Cache traffic by category: speculative, compulsory, capacity miss, conflict miss, write allocate, write back, coherency |
| | DRAM traffic |
| | I/O traffic |
| | % Utilization: cache controllers, memory controllers, I/O controllers |
| Computation | Instructions retired |
| | Instructions by type: floating point, integer, vector, load, store |
| | % Utilization: instructions retired per cycle |
| Energy | Energy per task for all components |
| | Time spent in each power state per component |

- Research Accelerator for Multiple Processors
- Manycore emulation on FPGAs
  - Using BEE3 boards
- Using RAMP to implement SHOT
  - Table to the lef100t shows the counters we have currently implemented
  - We are working on showing the benefit of the system by implementing an adaptive stack
    - Profile application resource usage
    - Dynamically adjust allocation of cores for lower energy

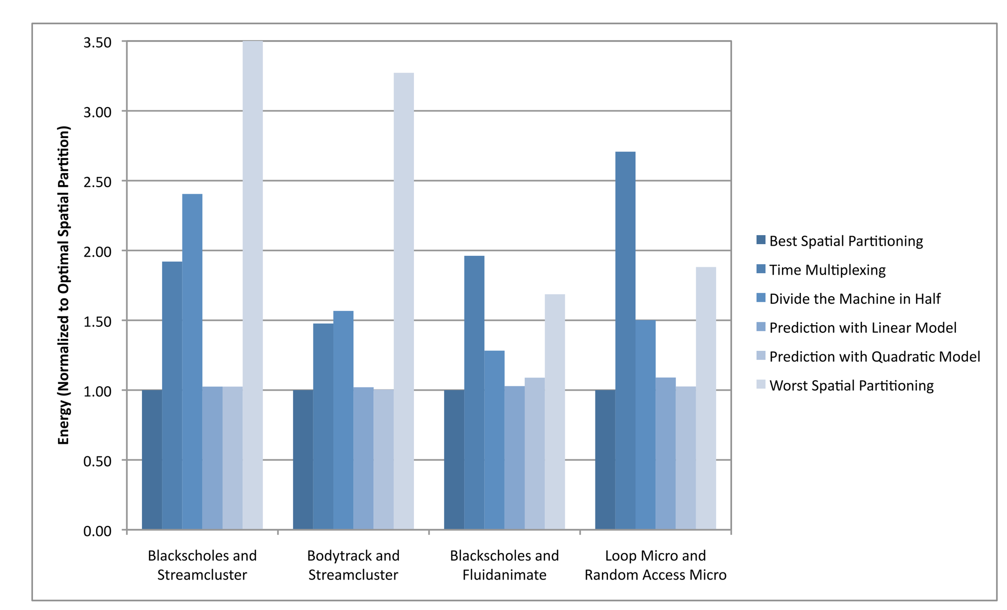



## Potential in Other Areas




- Standardized = Portable Software
- Autotuning
  - Prune search space
  - ML + Autotuning techniques (K. Datta and A. Ganapathi)
- Modeling
  - Performance
    - Automatically generate roofline model (S. Williams and A. Waterman)
  - Energy
- Distributed and Cloud Computing
  - Collect hardware performance data on a per request basis
    - Integrate with a system like X-Trace
    - Predict performance of Hadoop workloads using ML (S. Bird and A. Ganapathi)
  - Feedback to hardware designers

## Scheduling Experiments




- Using SHOT on RAMP
  - Running ROS
  - PARSEC Benchmarks
- SHOT data collected and used to make a simple energy model
- Use model for scheduling decisions in ROS
- With SHOT we are within 5% of optimal every time

## Isn't this a lot of hardware?

- We have more transistors available
  - The counters can be made low power and small
- Could Approximate
- The hardware cost isn't very high
  - SiCortex has 6 counters per core and over 3900 events
  - Only 0.05% of the Chip Area
- The real cost is verification
- It's worth the cost
  - Productive programming
  - Efficient execution

## Conclusions

- Performance is important and performance portability doesn't exist
  - Applications must be optimized for performance on each platform
  - It's too expensive to hand optimize every application for every platform
  - Environment changes depending on other applications running concurrently
  - Must have an adaptive stack that can use runtime information to adjust applications
- Scheduling Experiments show the potential of using SHOT information in the OS
  - Using SHOT is much lower energy that time-multiplexing or other baslines
  - It's within 5% of the optimal space partition every time