**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

**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

**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

**Our Implementation using RAMP**

- 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

**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

**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 baselines
- It’s within 5% of the optimal space partition every time