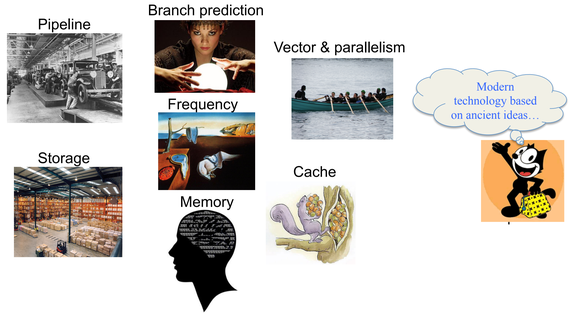
# Raexplore: Enabling Rapid, Automated Architecture Exploration for Full Applications

Yao Zhang, Prasanna Balaprakash, Jiayuan Meng*, Vitali Morozov, Scott Parker, and Kalyan Kumaran

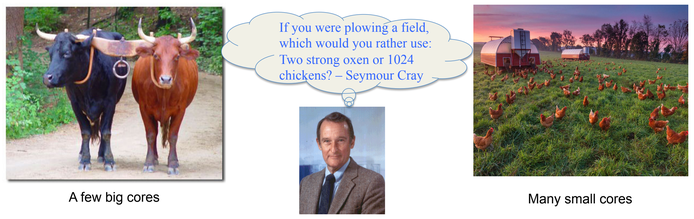
Leadership Computing Facility, Argonne National Laboratory, Argonne, IL (* Now affiliated with Google Inc.)

## Motivation

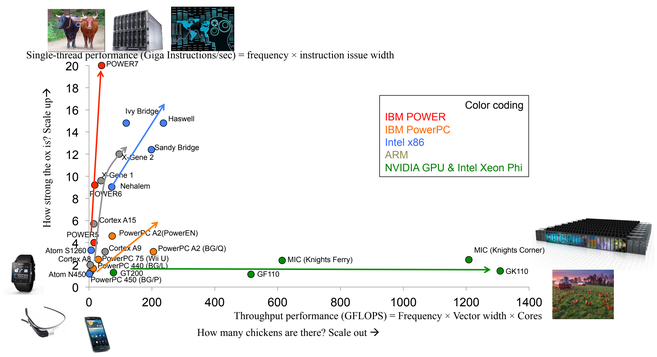
### Techniques used by modern computers



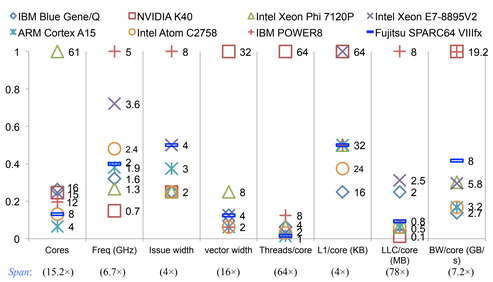

### Is it an art or a science?




### Processor landscape and trends




### Mutli-dimensional processor design space




### Research question: how should each processor evolve?

(to meet the needs for performance, energy efficiency, and emerging applications)

## Solution

### Goal: bring more "science" flavor to computer design

The current practice of computer design depends a lot on experts' experience, and often could be more of an art than a science, due to the complex nature of the problem. Our work aims to bring more "science" flavor into this process. Or, put in it mathematical terms:

$$\underset{x}{Minimize} \quad \text{programExecutionTime}(x_1, x_2, ..., x_n)$$

$$subject\ to \quad chipArea(x_1, x_2, ..., x_n) < c_{area}$$

$$energy(x_1, x_2, ..., x_n) < c_{energy}$$

$$Where\ x_1, x_2 ... x_n \text{ are a set of architecture design parameters}$$

### Existing methods: "three legs of science" in computer design

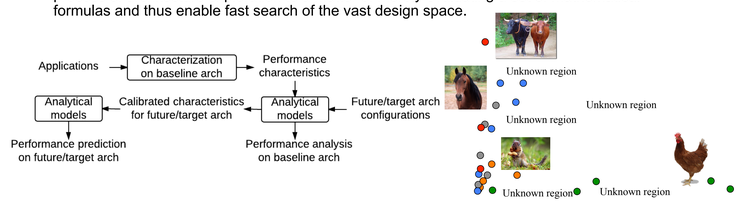
- Experiment (not predictive)
  - Run and measure a program on existing hardware; compare different hardware
- Simulation (slow)
  - Mimic the working mechanism of a future hardware
- Analytical modeling (inaccurate)
  - Use mathematical equations to describe the relation between design parameters and performance

### Our proposed hybrid methodology:

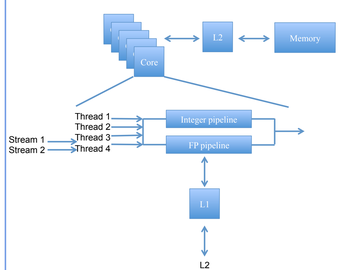
**experimental + analytical modeling = accurate + fast design exploration**

| Methodology | Speed | Accuracy | Large applications | Future architecture |
|---|---|---|---|---|
| Experiment | ✔ | ✔ | ✔ | ✖ |
| Simulation | ✖ | ✔ | ✖ | ✔ |
| Analytical model | ✔ | ✖ | ✖ | ✔ |
| Experimental + analytical | ✔ | ✔ | ✔ | ✔ |

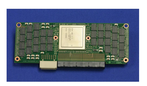
- We use experiments on baseline architecture and analytical performance models to predict the performance on future architecture. The diagram below describes our performance modeling methodology. The performance characteristics are hardware events measured by hardware-counter-based tools such as IBM HPM and Intel VTune.
- In other words, we use analytical models to explore unknown regions from known samples (color dots in the figure below) in the architecture design space. Analytical models reduce the process of architecture exploration to a matter of merely evaluating a set of mathematical formulas and thus enable fast search of the vast design space.




### Abstract computer model and design parameters




| Processor | Xeon Phi | BGQ |
|---|---|---|
| Frequency (GHz) | 1.2 | 1.6 |
| Cores | 60 | 16 |
| Int pipe latency (cycle) | 3 | 3 |
| FP pipe latency (cycle) | 4 | 5 |
| Max threads/core | 4 | 4 |
| Inst streams/thread | 2 | 1 |
| L1 size (KB), latency (cycle) | 32, 3 | 16, 3 |
| L2 size (MB), latency (cycle) | 31, 23 | 16, 42 |
| L2 linesize (bytes) | 64 | 128 |
| Mem bw (GB/s), latency (cycle) | 178, 750 | 28, 213 |



### Analytical performance models

Application-level model

$$timeTotal = \sum_i timeCodeBlock_i$$

$$timeCodeBlock = timeInst + timeMem - timeOverlap$$

Instruction pipeline model

$$timeInst = \frac{numInst}{instPerCycle}$$

$$instPerCycle = \frac{ILP}{avgInstLatency}$$

$$avgInstLatency = \frac{latencyInteger \times numIntegerInst + latencyFP \times numFPInst}{numIntegerInst + numFPInst}$$

$$ILP = ILP_{baseline} + threadsPerCore - threadsPerCore_{baseline} + streamsPerThread - streamsPerThread_{baseline}$$

Memory model

$$timeMem = \max(timeLatency, timeBandwidth)$$

$$timeBandwidth = \frac{(loads + stores) \times lineSizeLLC}{bandwidth}$$

$$timeLatency = \frac{numAccess}{accessPerCycle}$$

$$accessPerCycle = \frac{MLP}{avgMemLatency}$$

$$avgMemLatency = \frac{\sum latencyLevel_i \times numAccessLevel_i}{numAccess}$$

$$MLP = MLP_{baseline} + (ILP - ILP_{baseline}) \times accessPerInst$$

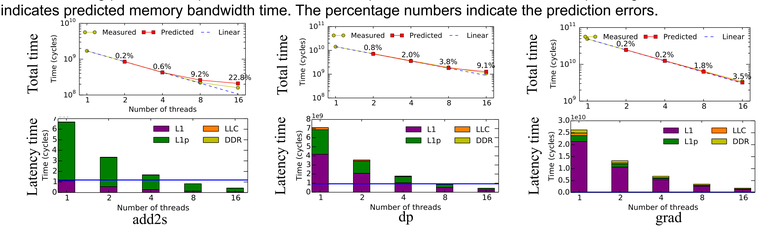
$$missRate = missRate_{baseline} \times \left(\frac{cacheSize}{cacheSize_{baseline}}\right)^{-0.5}$$

## Results

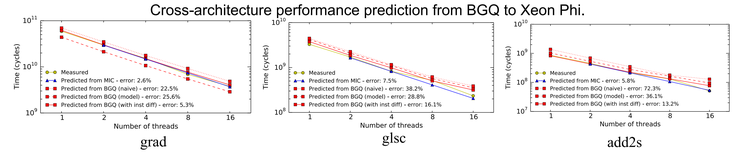
### Model Validation

Prediction errors within 3%-23% in our preliminary validations for various code blocks from the application Nekbone.

Threads scaling performance prediction on the BGQ chip for three code blocks: add2s, dp, and grad. The blue indicates predicted memory bandwidth time. The percentage numbers indicate the prediction errors.

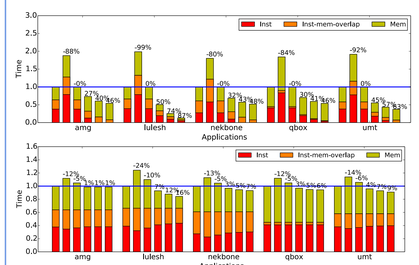



Cross-architecture performance prediction from BGQ to Xeon Phi.




### Architecture Exploration

(for a hypothetical BGQ chip on the CORAL applications, which is developed according to DOE's mission needs)

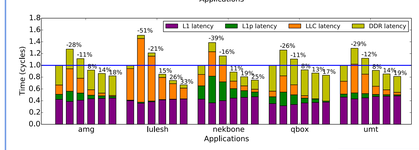



Core scaling. For each application, the 6 columns are respectively for baseline, 0.5, 1, 2, 4, 8, where the numbers represent scaling factors for #cores. The number above a column is the runtime difference from the baseline. Blue line indicates measured time for the baseline processor.
Finding: balanced design for #cores; if increased, cache/bw should keep up.




L1 cache scaling (total execution time). For each application, the 6 columns are respectively for baseline, 0.25, 0.5, 2, 4, 8, where the numbers represent scaling factors for the L1 cache size.

Finding: L1 could be bigger, although performance impact is not very significant.

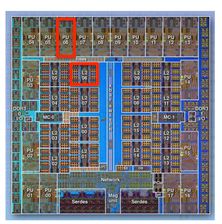



L1 cache scaling (memory latency time).
Finding: latency time is very sensitive to the L1 size, but with diminishing returns of larger cache size. LULESH sees the most benefit of a larger cache size, while AMG the least.

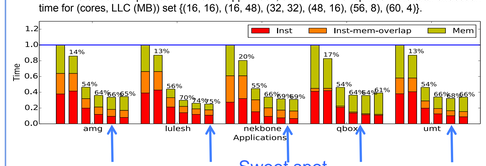
Case study for chip-area constraint-based architecture exploration: tradeoff between cores and L2 cache

- Hypothetical scenario
- 2× transistors
- 4× L1 size
- 3× bandwidth (e.g. stacked memory)
- Question: more cores or more L2 cache?
- Finding: L2 seems overdesigned. Having more cores are more beneficial than L2.

| Keep total transistors constant | | |
|---|---|---|
| | Cores | L2 Cache (MB) |
| Baseline | 16 | 16 |
| Hypothetical next-generation BGQ: | | |
| Option 1 | 16 | 48 |
| Option 2 | 32 | 32 |
| Option 3 | 48 | 16 |
| Option 4 | 56 | 8 |
| Option 5 | 60 | 4 |



Core-LLC tradeoff exploration. For each application, the 5 columns represent the total execution time for (cores, LLC (MB)) set {(16, 16), (16, 48), (32, 32), (48, 16), (56, 8), (60, 4)}.




Caveat:

- This case study is for the purpose of demonstrating our methodology/model.
- The findings are specific to our applications.
- In reality, performance is constrained by power consumption. Cores use more power than cache and having too many cores will exceed the designed power limit.
- Future work will incorporate power model as a design constraint in addition to area constraint.

### Summary and Future Work

Summary:

- Propose a novel experimental + analytical modeling methodology for architecture exploration;
- Build analytical performance models for BGQ and Xeon Phi for case studies;
- Preliminary validation of our approach and models targeting real-world large applications for same-architecture and cross-architecture performance prediction;
- Explore and suggest architecture scaling options for BGQ.

Future work:

- Include multicores and GPUs in our study;
- Incorporate models for resource constraints including chip area and power;
- Integrate system-level models for network, I/O, and storage;
- Introduce program transformation parameters to the optimization space for hardware and software co-design.

**Acknowledgements** This work is supported by the U.S. Department of Energy under contract DE-AC02-06CH11357 and the LDRD X-PECT project 2013-213-NO. The research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.