

Using Auto-tuning to Generate Optimal Code for Multicore

Shoaib Kamil and Kaushik Datta
ParLab Grand Opening
December 1, 2008

❖ Problems:

- Multicore revolution has produced a wide variety of architectures
- Compilers alone fail to fully utilize system resources
- May need to tune for every architecture/compiler/dataset combination
- Programmer time is expensive

❖ Solution: Auto-tuning

- Identify set of relevant optimizations for a given kernel
- Choose a reasonable parameter range for each optimization
- *Automatically* search parameter space for each new architecture/dataset

❖ Benefits:

- Same code can be tuned across variety of architectures/compiler/datasets
- Code should scale to new architecture generations (w/ more cores)
- Code can be tuned for a variety of relevant metrics
- Programmer tuning time is minimized
- Proven track record (e.g., ATLAS, SPIRAL, FFTW, OSKI)

❖ Drawbacks:

- Code needs to be rewritten for new programming models (cache-based vs. Cell vs. GPUs, portable C vs. x86 SSE, etc.)
- Tuning time can vary drastically based on search space traversal

- ❖ How do we:
 - traverse the parameter space (search, heuristics, something else)?
 - know when to stop tuning?
 - handle poor compilers?
 - auto-tune a composition of two or more kernels?
 - auto-tune for a loaded system?
 - weight relevant metrics (performance, power efficiency, productivity, memory usage, etc.)?
- ❖ Could we devise a universal auto-tuner (across all kernels)?