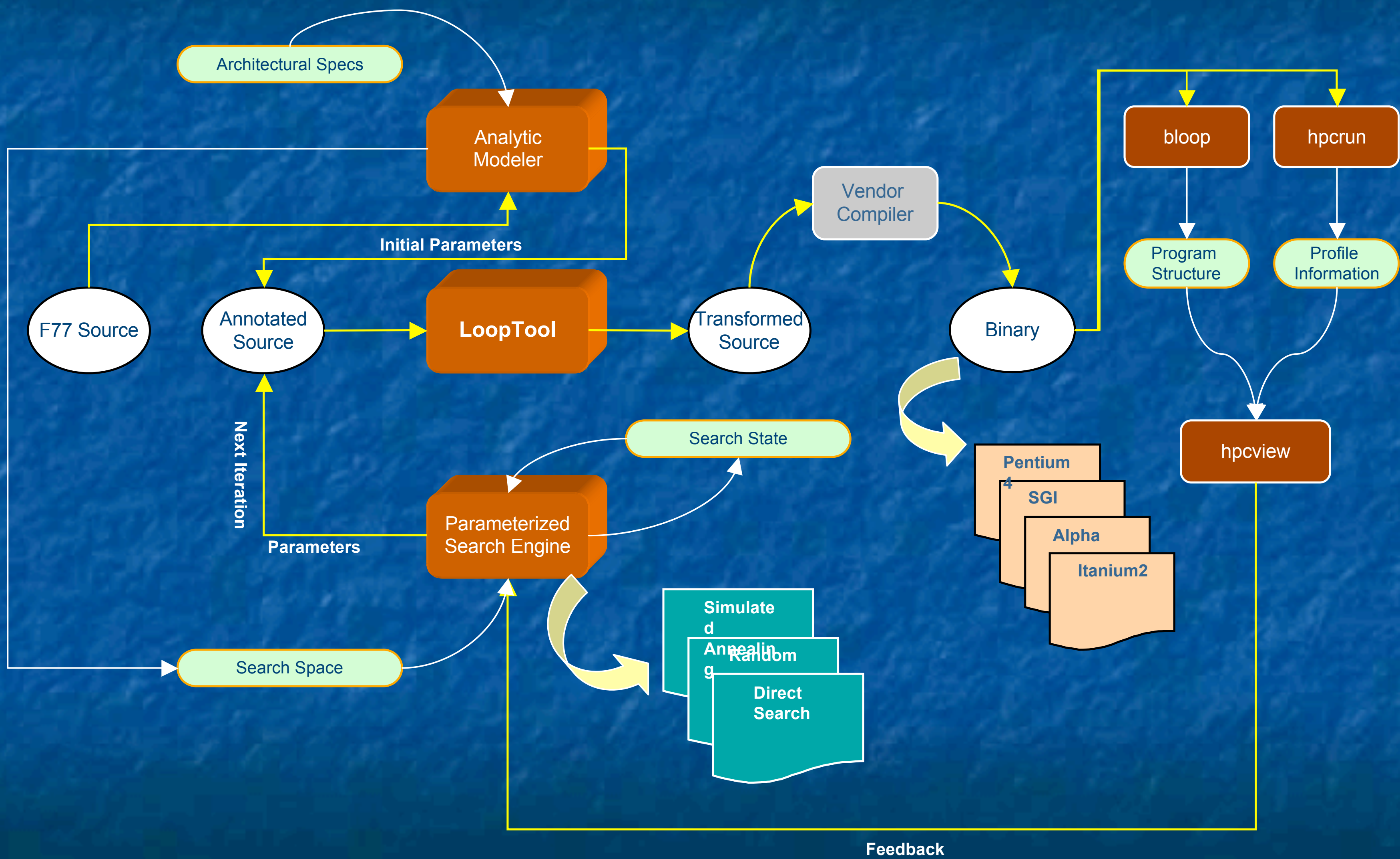
Using Cache Models and Empirical Search in Automatic Tuning of Applications

Apan Qasem   Ken Kennedy   John Mellor-Crummey
Rice University
Houston, TX

Outline

- Overview of Framework
  - Fine grain control of transformations
  - Loop-level feedback
  - Search strategy
- Experimental Results
- Empirical Tuning of Loop Fusion
- Future Work

A Framework for Autotuning



Fine Grain Control of Transformations

- Not enough control over transformations in most commercial compilers
- Example `sweep3d`
  - 68 loops in the `sweep` routine
  - Using the same unroll factor for all 4 loops results in a 2% speedup
  - Using 4 different unroll factors for 4 loops results in a 8% speedup

Fine Grain Control of Transformations

- LoopTool: A Source to source transformer
  - Performs transformations such as loop fusion, tiling, unroll-and-jam
  - Applies transformations from source code annotation
  - Provides loop level control of transformations

```
cdir$ unroll 4
do j = 1, N
   cdir$ block 16
   do i = 1, M
      cdir$ block 16
     do k = 1, L
        S1(k, i, j)
      enddo
   enddo
 enddo
```

Feedback

- Feedback beyond whole program execution time
  - Need feedback at loop level granularity
    - Can potentially optimize independent code regions concurrently
  - Need other performance metrics:
    - Cache misses
    - TLB misses
    - Pipeline stalls

Feedback

- HPCToolKit
  - `hpcrun` – profiles executions using statistical sampling of hardware performance counters
  - `bloop` – retrieves loop structure from binaries
  - `hpcview` – correlates program structure information with sample-based performance profiles

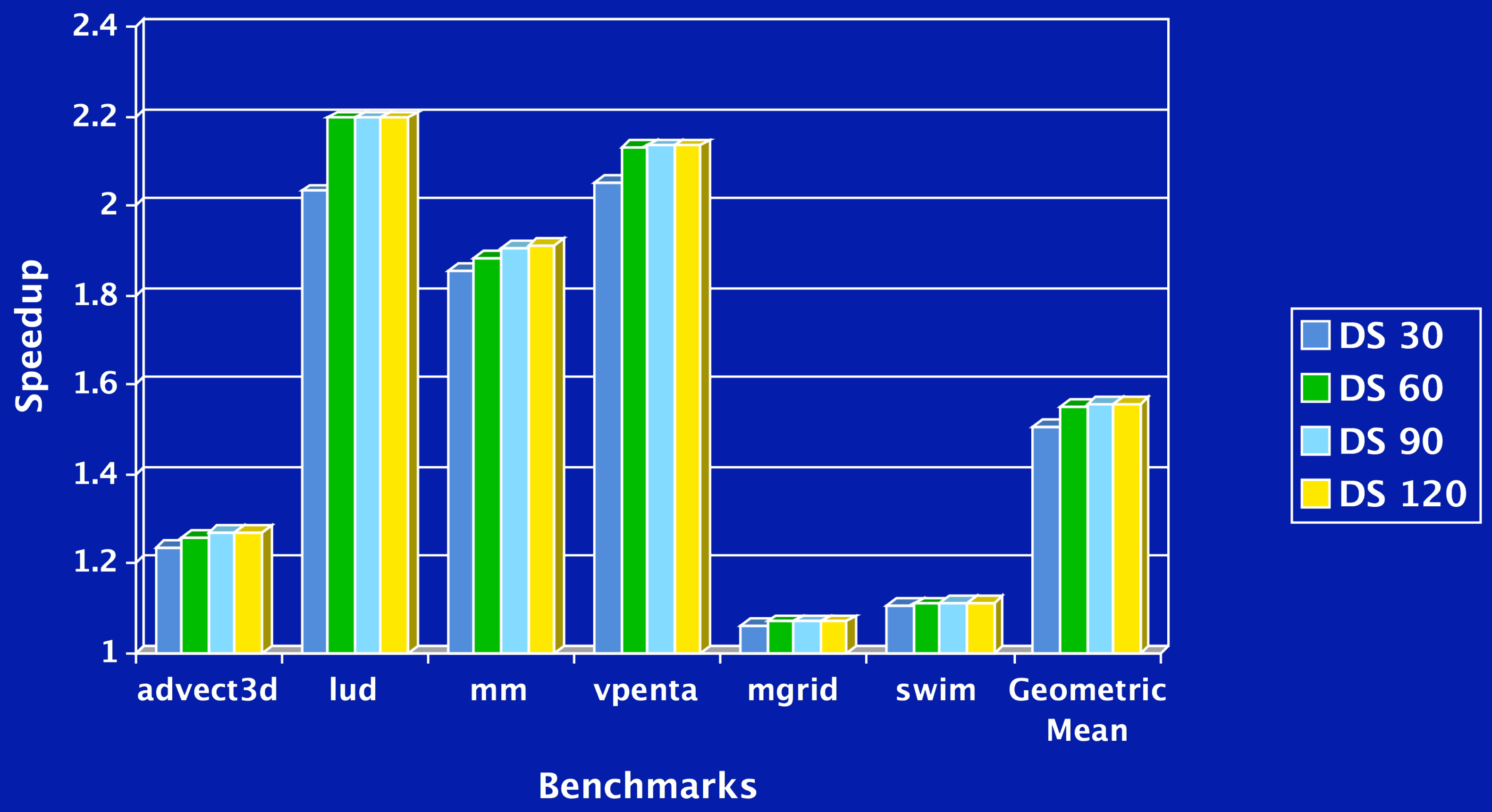
Search Space

- Optimization search space is very large and complex
  - Example mm
    - 2-level blocking (range 1-100)
    - 1-unrolled loop (range 1-20)
    - 100 x 100 x 20 = 200,000 search points
  - Characteristics of the search space is hard to predict
  - Need a search technique that can explore the search space efficiently

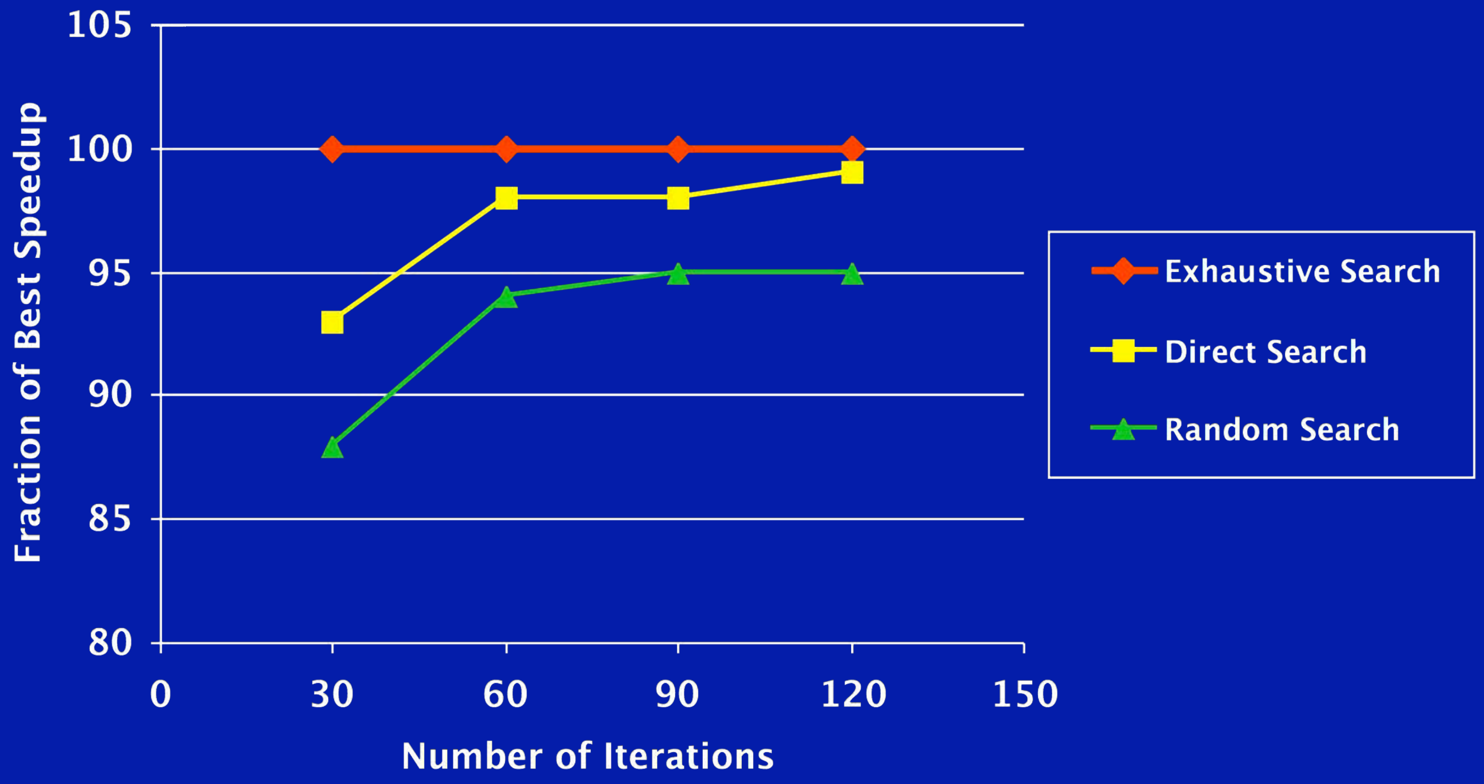
Search Strategy

- Pattern-based direct search method
  - The search is derivative-free
    - Works well for a non-continuous search space
  - Provides approximate solutions at each stage of the calculation
    - Can stop the search at any point when constrained by tuning time
  - Provides flexibility

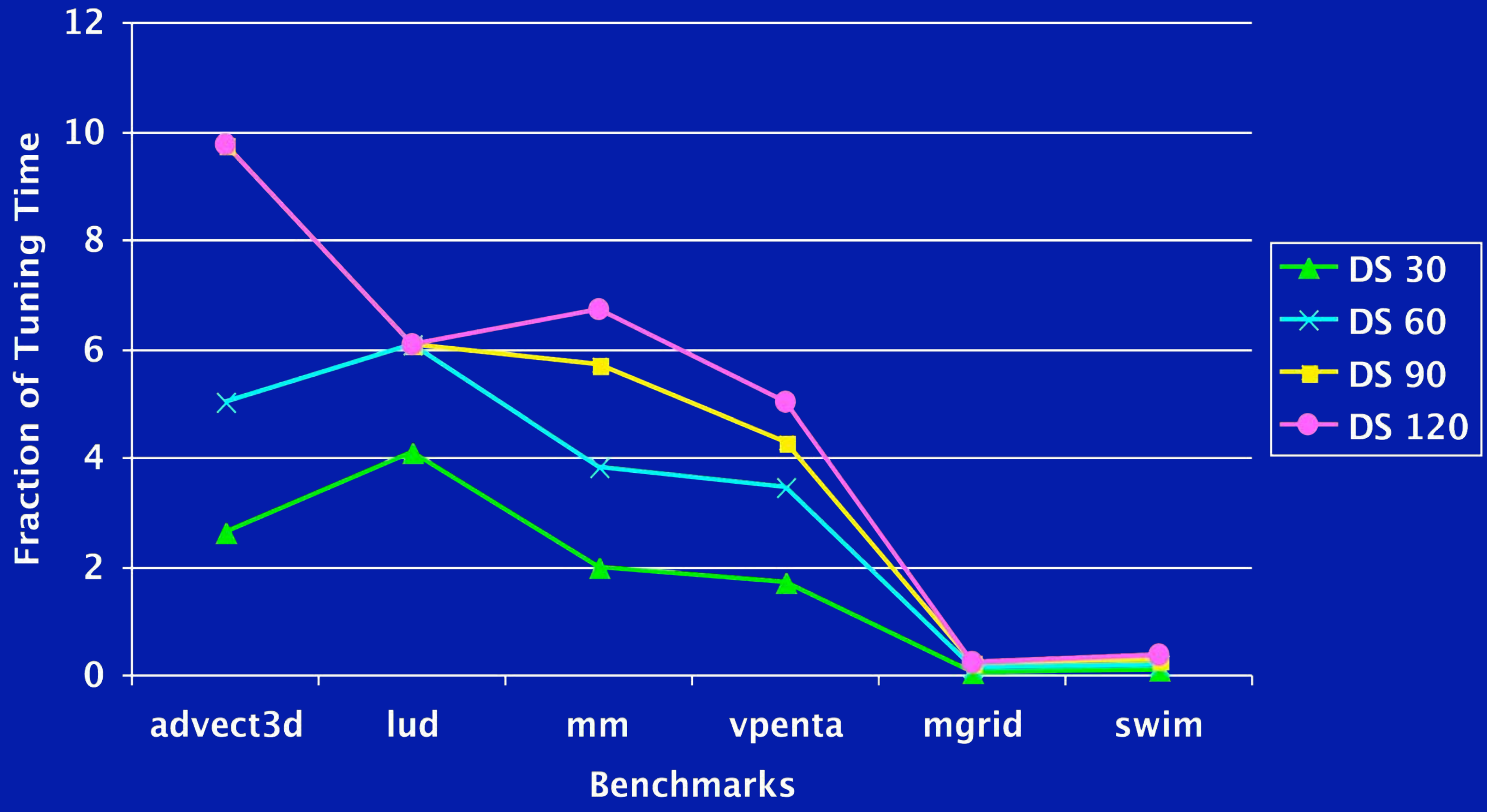
Performance Improvement



Performance Improvement Comparison



Tuning Time Comparison



Empirical Tuning of Loop Fusion

- Important transformation when looking at whole applications
- Poses new problems
  - Multi-loop reuse analysis
  - Interaction with tiling
  - What parameters to tune?
    - Trying all combinations is perhaps too much

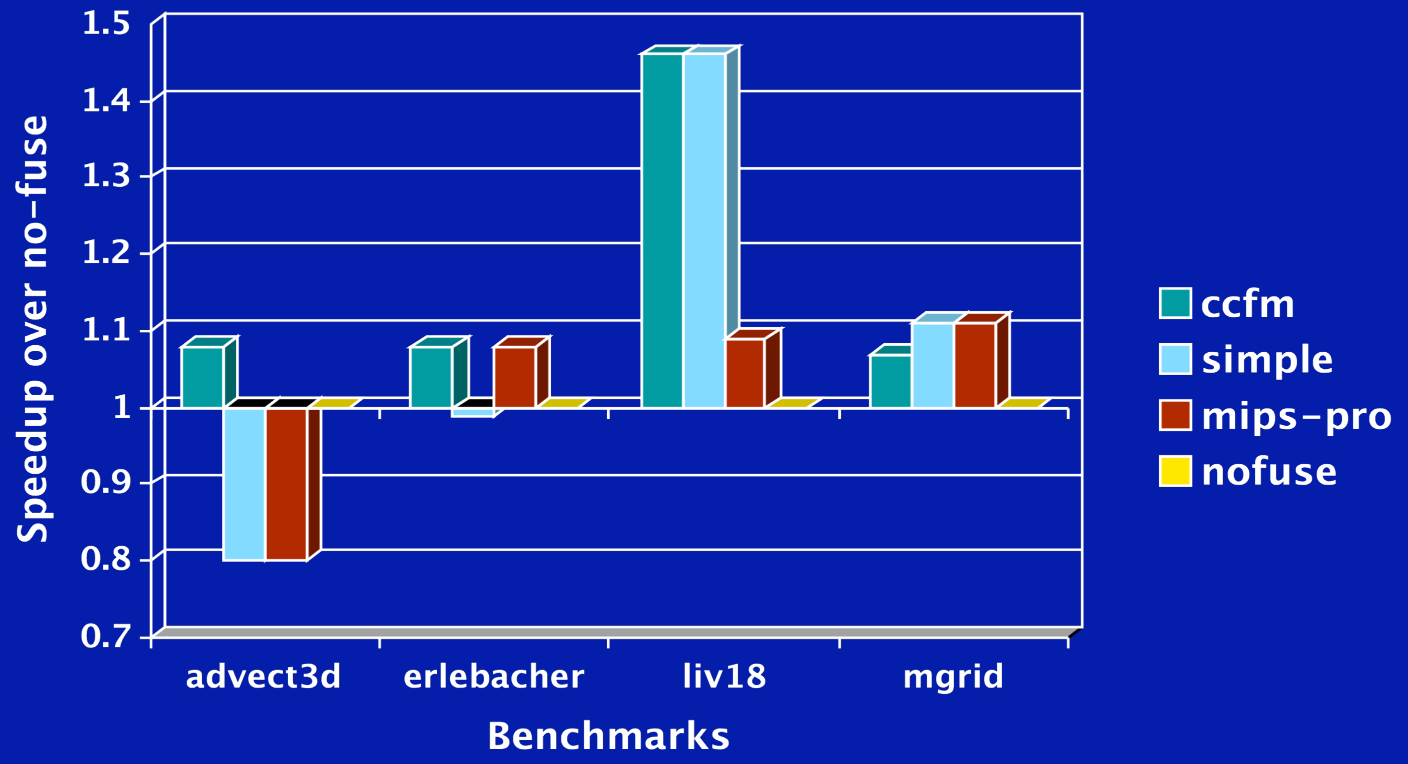
Profitability Model for Fusion

- Hierarchical reuse analysis to estimate cost at different memory levels
- A probabilistic model to account for conflict misses
  - Effective Cache Size
- Resource constraints
  - Register Pressure
  - Instruction Cache Capacity
- Integrate the model into a pair-wise greedy fusion algorithm

Tuning Parameters for Fusion

- Use search to tune the following parameters
  - Effective Cache Size
  - Register Pressure
  - Instruction Cache capacity
- Preliminary Experiments
  - Works reasonably well

Speedup from Loop Fusion



Summary

- Autotuning framework able to improve performance on a range of architectures
- Direct search able to find *good* tile sizes and unroll factors by exploring only a small fraction of the search space
- Combining static models with search works well for loop fusion

Future Work

- Refine cache models to capture interactions between transformations
- Consider data allocation strategies for tuning

Extra Slides Begin Here

Platforms

Processor	L1	L2	L3	Compiler
Alpha 21264A @ 667MHz	8K	8M	-	Compaq Fortran V5.5-1877
Itanium2 @ 900 MHz	16K	256K	1.5M	Intel Fortran Compiler 7.1
SGI Origin R10K @ 195 MHz	32K	1M	-	MipsPro 7.3
Pentium 4 @ 2 GHz	8K	512K	-	Intel Fortran Compiler 7.1

Performance Improvement for Different Architectures