

On the Performance of Personal Codes

John Feo
SUN Microsystems
San Diego SuperComputer Center

Simulation drives scientific research

- Computer simulation is an essential tool in all scientific fields
- The best research involves state-of-the-art applications that are compute and/or memory bound
- **Efficiency and parallelism are essential**
- A faster, more parallel program can do bigger and better science
 - more results
 - bigger problems
 - better accuracy

Three classes of codes

- ISV codes
- Community codes
- Personal codes

ISV codes

- Commercial scientific codes
 - Nastran
 - Dyna3D
 - Gaussian
- Large, mature codes maintained and optimized by ISV and computer vendors
- Significant performance improvements require algorithm or system platform changes
 - 5% - 10% improvements via code changes

Community codes

- Scientific research codes developed and maintained by a research community
 - Blast
 - Amber
 - Clustalw
- Typically, smaller and less mature than ISV codes
- Performance improves over time via
 - faster platforms
 - better algorithms
 - code optimization (maybe up to 50%)

Personal codes

- Developed by individual researchers for their personal use
- Small, immature programs written primarily in Fortran or C
- Surprisingly, well written
 - modular
 - commented
 - most have make environment
- Most are running many times slower than they should and almost none are parallel

Typical errors

- Arrays are not traversed in storage order
- Loop bodies are too big
- Loop bodies are too small
- Nested loops are not unrolled and blocked
- Inner loops include function calls or conditional expressions
- Sets of constant values are recomputed, rather than computed once and looked up

Fast and parallel are difficult

- Complex memory hierarchies
 - L1, L2, ... caches
 - local memory
 - remote memory
- Parallelism
 - division of work
 - division of data
 - communication/synchronization
 - race conditions
- Increased compiler and tool complexity

Performance improvements

<u>#</u>	<u>Speedup</u>
1	15.5
2	6.3
3	5.8
4	5.0
5	3.3
6	2.8
7	2.5
8	2.2
9	2.1
10	2.0

Case Study 1

- Multiple alignment code
- Alignment score is dot product of profiles
 - $\text{Score}(i, j) = \text{DOT}(\text{profile1}[i], \text{profile2}[j])$
- Replaced vector dot products in inner loop with a matrix multiply outside the loop
 - $\text{Score} = \text{MM}(\text{profile1}, \text{profile2})$
- 2x improvement on one processor

Case study 2

- Protein database parser
- List of atoms scanned and the index of those atoms that satisfy the selection criteria saved for further processing
- Scan loop is in cache-order, process loops are not
- Saved parameters of selected atoms to temporary arrays, so that process loop would be stride one
- 6x improvement on one processor

Case study 3

- Neural net application
- Replaced 2-D array of arrays with 1-D structure
- Hoisted loop invariant pointer references
- Software pipelined all inner loops
- 2.5x improvement on one processor

Case study 4

- Genomics code
- Passed pointer to struct rather struct in all critical function calls
 - structs are passed by value
- Eliminated unnecessary reassignments of structures
 - rhs values are copied to lhs memory location
- **2x improvement on one processor**

Case study 5

- Multigrid code
- Inverted loops so wavefront progressed column-wise
- Added third dimension to value array
 - $V(\dots, 1) = \textit{new values}$
 - $V(\dots, 2) = \textit{old values}$
- Compressed coarse grids into continuous memory
 - outer iterative loop over each coarse grid
 - inner loop processes every n th elements, $n = \{1, 2, 5, 10\}$
- 15x improvement on one processor

Conclusions

- Most university research codes are **VERY inefficient** and **NOT parallel**
- Simple techniques can improve significantly the codes' performance and scalability
- **Funds** used to support the research and pay for the computer resources **are being wasted**
- The education community must find a way to eliminate this problem
 - Manuals and reference material
 - Short courses or workshops
 - Semester courses

Free help is available

- I am available to help you optimize and parallelize your code
- I need all source files, makefile, run command, and input/output files
- I usually turn the code back within 2 weeks
- Contact me by email at feo@sdsc.edu