REMEMBER TO SUBMIT YOUR EXAM TO STUDENTPORTALEN AT 26th of May, 23:59
Submission structure:

1) Unoptimized code (+make)
2) Optimized code (+make)
3) Tests
4) Report
HPC Libraries

- HPC languages
- Things to consider about libraries
- Sources of libraries
- Example libraries
HPC Languages

- C
- C++
- Java
- Python
- Julia
- Matlab
- R
## Language Performance

<table>
<thead>
<tr>
<th></th>
<th>Julia 3f670da0</th>
<th>Python 2.7.1</th>
<th>Matlab R2011a</th>
<th>Octave 3.4</th>
<th>R 2.14.2</th>
<th>JavaScript V8 3.6.6.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>fib</td>
<td>1.97</td>
<td>31.47</td>
<td>1336.37</td>
<td>2383.80</td>
<td>225.23</td>
<td>1.55</td>
</tr>
<tr>
<td>parse_int</td>
<td>1.44</td>
<td>16.50</td>
<td>815.19</td>
<td>6454.50</td>
<td>337.52</td>
<td>2.17</td>
</tr>
<tr>
<td>quicksort</td>
<td>1.49</td>
<td>55.84</td>
<td>132.71</td>
<td>3127.50</td>
<td>713.77</td>
<td>4.11</td>
</tr>
<tr>
<td>mandel</td>
<td>5.55</td>
<td>31.15</td>
<td>65.44</td>
<td>824.68</td>
<td>156.68</td>
<td>5.67</td>
</tr>
<tr>
<td>pi_sum</td>
<td>0.74</td>
<td>18.03</td>
<td>1.08</td>
<td>328.33</td>
<td>164.69</td>
<td>0.75</td>
</tr>
<tr>
<td>rand_mat_stat</td>
<td>3.37</td>
<td>39.34</td>
<td>11.64</td>
<td>54.54</td>
<td>22.07</td>
<td>8.12</td>
</tr>
<tr>
<td>rand_mat_mul</td>
<td>1.00</td>
<td>1.18</td>
<td>0.70</td>
<td>1.65</td>
<td>8.64</td>
<td>41.79</td>
</tr>
</tbody>
</table>

**Figure**: benchmark times relative to C++ (smaller is better).
function randmatstat(t)
    n = 5
    v = zeros(t)
    w = zeros(t)
    for i = 1:t
        a = randn(n,n)
        b = randn(n,n)
        c = randn(n,n)
        d = randn(n,n)
        P = [a b c d]
        Q = [a b; c d]
        v[i] = trace((P.'*P)^4)
        w[i] = trace((Q.'*Q)^4)
    end
    std(v)/mean(v), std(w)/mean(w)
end

Julia random matrix statistics benchmark.

Equivalent C++ benchmark
Remember the HPC priorities

1) Correctness
2) Flexibility
3) Performance
C++ vs C

- Templates vs typedefs, code duplication, and macros
- Classes & methods vs structs
- Really, it's the template libraries in C++
- And C maps closely to machine code
Downsides of C/C++

- Error-prone programming
- Huge and ugly code
  - Hard to maintain
  - Hard to move
  - Hard to communicate
- Portability must be done explicitly
Java as HPC language

- Just-In-Time Compiler (JIT) is smart.
- Cross-platform
- Easier to use than C/C++
- Growing availability of HPC libraries
- Up to 90% as fast as C/C++
- Performance portable?
Python for HPC

- Use Python to *coordinate* computation codes
- Lots of ties to C/C++ and Fortran
- SciPy and NumPy libraries

For interested students:  
http://skillsmatter.com/podcast/home/high-performance-python/js-1630
Julia

- Easy like Python, fast like C?
- JIT compiled
- Message-passing parallelism
- Cloud computing mode
- Built-in C & Fortran libs for e.g. linear algebra and FFT's
Julia

- Go to http://julia.forio.com/repl.htm
- Try typing plot(x->sin(x^2)/x, -2pi,2pi)
- Neat!
Matlab

- Uses Intel's MKL library for matrix operations
- Versatile but expensive packages
- Decent data plotting functions
- Very slow as a programming language
- Write your own computational kernels in an external language
The R Project for Statistical Computing

- Free software environment for statistical computing and graphics
- With Rcpp package:
  - Like Matlab, it can call C++
  - Supports “inline” C++ code
Matrix Multiply

<table>
<thead>
<tr>
<th>Method</th>
<th>PhP (ms)</th>
<th>Python (ms)</th>
<th>Python (Jitted) (ms)</th>
<th>Java (ms)</th>
<th>In C (ms)</th>
<th>Transposed (ms)</th>
<th>tiled (ms)</th>
<th>Vectorized (ms)</th>
<th>BLAS MxM (ms)</th>
</tr>
</thead>
</table>

Cycles/OP

343, 227, 20, 13, 1/2, 1/2, 1/3, 1/5, 1/17

6.172 Saman Amarasinghe, MIT Fall 2011
Babel

- High-Performance Language Interoperability
- Uses an Interface Definition Language to express software interfaces
- Babel generates glue code and skeletons
Things to consider about libraries

- Cost
- Licensing
- Performance
- Thread safety
- Stability
- Feature-completeness
- Update frequency
- Generality
- Interface

Copyright Candida Höfer
• Some scientific computing libraries are all-purpose
• Typically, the more specialized, the more up-to-date
• But all-purpose libraries can be a good first effort
Sources:

- nr.com, a classic source of source-code
- Older editions in C and Fortran are free online
- A great go-to place for basic and understandable algorithms
- Not so advanced, not always fast
Sources: the Netlib

- Netlib.org
  - A huge repository: http://www.netlib.org/master/expanded_liblist.html
- Maintained by AT&T, Bell Labs, U Tennessee, Oak Ridge Nat'l Lab
- It is OLD: Mostly Fortran code
- Contains among others:
  - BLAS
  - EISPACK
  - LAPACK
Sources: GAMS

- Guide to Available Mathematical Software
- http://gams.nist.gov/
- Indexes Netlib and a set of commercial packages
- Provides a decision-tree for finding useful functions
• Truly general-purpose C++ libraries
• Over 80 peer-reviewed libraries
• Includes linear algebra, pseudorandom numbers, graphs, logic, data structures, ...
• Heavy use of templates
• Most libraries are header based and don't need compilation
• Several Boost libraries are headed for incorporation into standard C++
What about Boost performance?

- Boost reports comparisons with R with mixed results

Should you use Boost?

- “Use of [Boost] encourages people to fit a solution to a problem rather than finding a solution to a problem. The solution should always be appropriate for the data at hand and the constraints of the hardware, etc. Boost has an extremely narrow view of the "world" and its appropriate use is limited. Really, I discourage it because it leads programmers down the wrong direction right away, I often say if you feel like you need [Boost] you probably don't really understand the problem that you're trying to solve.” - Mike Acton (Engine Director at Insomniac Games)
• Commercial product
• Large collection of mathematical and statistical algorithms
• Wide range of applicability: from Excel in Windows to Matlab in Linux
• Think “R but in a really fast library”
• Gnu Scientific Library
• Numerical library for mathematical routines
  • BLAS (through e.g. ATLAS)
  • Monte Carlo Integration
  • Differentiation
  • Differential Equations
  • Simulated Annealing
  • ...
• Distributed under GPL license.
Scipy

- Open-source
- Depends on NumPy
  - Fast N-dimensional arrays
  - Basic linear algebra
  - Basic Fourier transforms
- Reasonably wide range of functionality
Basic Linear Algebra Subprograms
Linear Algebra Package

- Lots of problems are solved in a linearized form
- Linear algebra software is incredibly well-optimized
- There are many BLAS packages
- They all have the same functions and interface
- How to decide which is right?
  - Intel MKL may be the best
  - ATLAS is free and is portable and is tunable
  - PLASMA – multicore
  - MAGMA - GPU
• Using a BLAS or LAPACK library:

• Different routines exist for most ordinary operations

• Usually must choose between:
  • Dense or sparse?
  • Single, double, complex?
  • Transpose or not?

• Think about if:
  • You want to take advantage of complicated structure in matrix
  • Preconditioning is necessary
  • The order you do the operations is important
• Commercial product
• Probably the fastest in general
• Math Kernel Library
• Optimized for Intel processors
• C & Fortran
• Includes BLAS, LAPACK, also sparse solvers, FFT,..
GotoBLAS

- Discontinued project at U of Texas
- MKL-like speeds in many benchmarks
- Goto is the developer's last name, not “go to”
- Still in use on top supercomputers
- Hand-optimized assembly routines!?

Eigen

- Open Source
- Template Library for linear algebra
- Supports dense and sparse matrices
- Supports complex numbers
- Various matrix decompositions
- Good online documentation
- Quite fast!
ATLAS

- Automatically Tuned Linear Algebra Software
- In Netlib, free
- Speed of BLAS3 routines comparable to MKL's
- Can be compiled as threaded or non-threaded
  - If you do your own threading, link to the serial interface and ATLAS' threading will be turned off.
- Recommended to use “architectural defaults”
  - Tuning for best performance is difficult and seldom necessary.
Some benchmarks

MTL

- Matrix Template Library
- Open Source, “Supercomputing”, or GPU editions
- Supports dense and sparse matrices
- Enables natural expression of matrices in a C++ environment
- Performance not stellar, but not bad at all
• So you have some data.
• What does it look like?
• A big pile of numbers looks like nothing at all.
• Open-source
• Wrapper for OpenGL
• Represent large datasets (like a foot)
• Classes for things like molecules
MathGL

- GPL license
- Interfaces with:
  - OpenGL
  - Qt
  - FLTK
- 55 general types of graphics
- Can be used from many languages and platforms
MathGL

- GPL license
- Interfaces with:
  - OpenGL
  - Qt
  - FLTK
- 55 general types of graphics
- Can be used from many languages and platforms