Building R for Speed using the Intel MKL library

Jonathan Boyle

Motivation

- I work in IT Services at University of Manchester
  - Supporting research software
    - e.g. training, consultancy, code optimisation
- I want to learn how to use R
  - At least sufficiently well to help novice users
    - Recently several requests for help speeding up R
- My first step is to install R, but what is optimal?
  - Precompiled R? Or compile the source code?
    - If I compile what's the best method to get speed?
  - Is R multithreaded?
    - No
- Also how does R compare to MATLAB for speed?
  - e.g. for which cases should I recommend R?

Devise linear algebra tests

- Select 2 array sizes
  - 'Small' array has 100 x 100 entries
  - 'Large' array has 5000 x 5000 entries
- Matrix-matrix multiply and subtract

  i.e. **`mat = mat %*% mat2 - mat2`**
  - x200000 for 'small' arrays (test: **SMM**)
  - x10 for 'large' arrays (test: **LMM**)
- Linear solver

  i.e. **`sol = solve(mat,rhs)`**
  - x20 for 'large' arrays (test: **SOL**)

And tests of raw compute

- Element-wise multiplication and subtraction

  i.e. **`mat = mat * mat2 - mat2`**

  – x1000000 for ‘small’ arrays (test: **SEM**)

  – x200 for ‘large’ arrays (test: **LEM**)

Data for precompiled R

- R version 3.0.2
  - This is the precompiled version installed by Ubuntu 12.04
  - Times are measured in seconds
  - PC is Dell Precision T1650 with Intel Core i7-3770 CPU

	Precompiled R (4 threads)	Precompiled R (1 thread)	MATLAB R2012a (4 threads)	MATLAB R2012a (1 thread)
SEM	20	21	8	8
LEM	28	29	6	7
SMM	119	119	11	20
LMM	180	183	25	88
SOL	142	142	12	33

Data for precompiled R

- Element-wise multiplication
  - R is slower than MATLAB
  - MATLAB speed stays same for 1 & 4 threads
- Linear algebra
  - R much slower than multithreaded MATLAB
  - R significantly slower than single thread MATLAB
- So can I speed up R using
  - Intel MKL maths libraries?
  - Intel compilers?

Linear algebra libraries

- When compiling R we have a choice
  - Either use the default R BLAS and LAPACK libraries
  - Or link against optimised versions e.g.
    - Intel® Math Kernel Library (MKL) - commercial
    - Or use free versions e.g. OpenBLAS
- Fast linear algebra often essential for speed
  - BLAS = Basic Linear Algebra Subprograms
  - LAPACK = Linear Algebra PACKage
- MKL generally considered the fastest
- Anyone tried Revolution Analytics' R Enterprise?
  - Compiled using MKL
  - Free for students, teachers and researchers

Compiled R (GNU compilers)

	GNU compilers without MKL (4 threads)	GNU compilers without MKL (1 thread)	GNU compilers with MKL (4 threads)	GNU compilers with MKL (1 thread)
SEM	20	20	20	20
LEM	29	28	28	28
SMM	124	125	23	32
LMM	**679**	**670**	27	89
SOL	**386**	**385**	23	68

- Note
  - GNU compilers v4.6.3
  - Compiled using -O3 optimisation
  - Similar data obtained using Intel compilers

Compiled R (GNU compilers)

- Linear algebra in R much faster using MKL
  - But other than LMM still slower than MATLAB
    - Why?
- For LMM and SOL
  - Compiled R without MKL significantly slower than my precompiled Ubuntu R
    - Presumably precompiled R uses some open source BLAS?
- Element-wise operations don't use BLAS
  - So no speed up using MKL
- **Q: How representative is this of real R code?**

Recap: Best R vs MATLAB

	MATLAB R2012a (4 threads)	MATLAB R2012a (1 thread)	GNU compilers with MKL (4 threads)	GNU compilers with MKL (1 thread)
SEM	8	8	20	20
LEM	6	7	28	28
SMM	11	20	23	32
LMM	**25**	**88**	**27**	**89**
SOL	12	33	23	68

- For LMM times are comparable
  - But where is R spending time in other tests?

Precompiled R for Windows

- Precompiled R running in Windows 7
  - Download from cran.r-project.org

	Windows precompiled R	Ubuntu precompiled R	Ubuntu compiled R using GCC without MKL
SEM	20	20	20
LEM	32	28	29
SMM	115	119	124
LMM	**716**	180	679
SOL	**386**	142	386

- Conclude
  - Linux precompiled uses optimised BLAS (not MKL)
  - Windows precompiled uses the default R BLAS