The Evolving Hardware Landscape and the Implications for Libraries

Mike Dewar

Date of presentation

nag®

Experts in numerical algorithms and HPC services

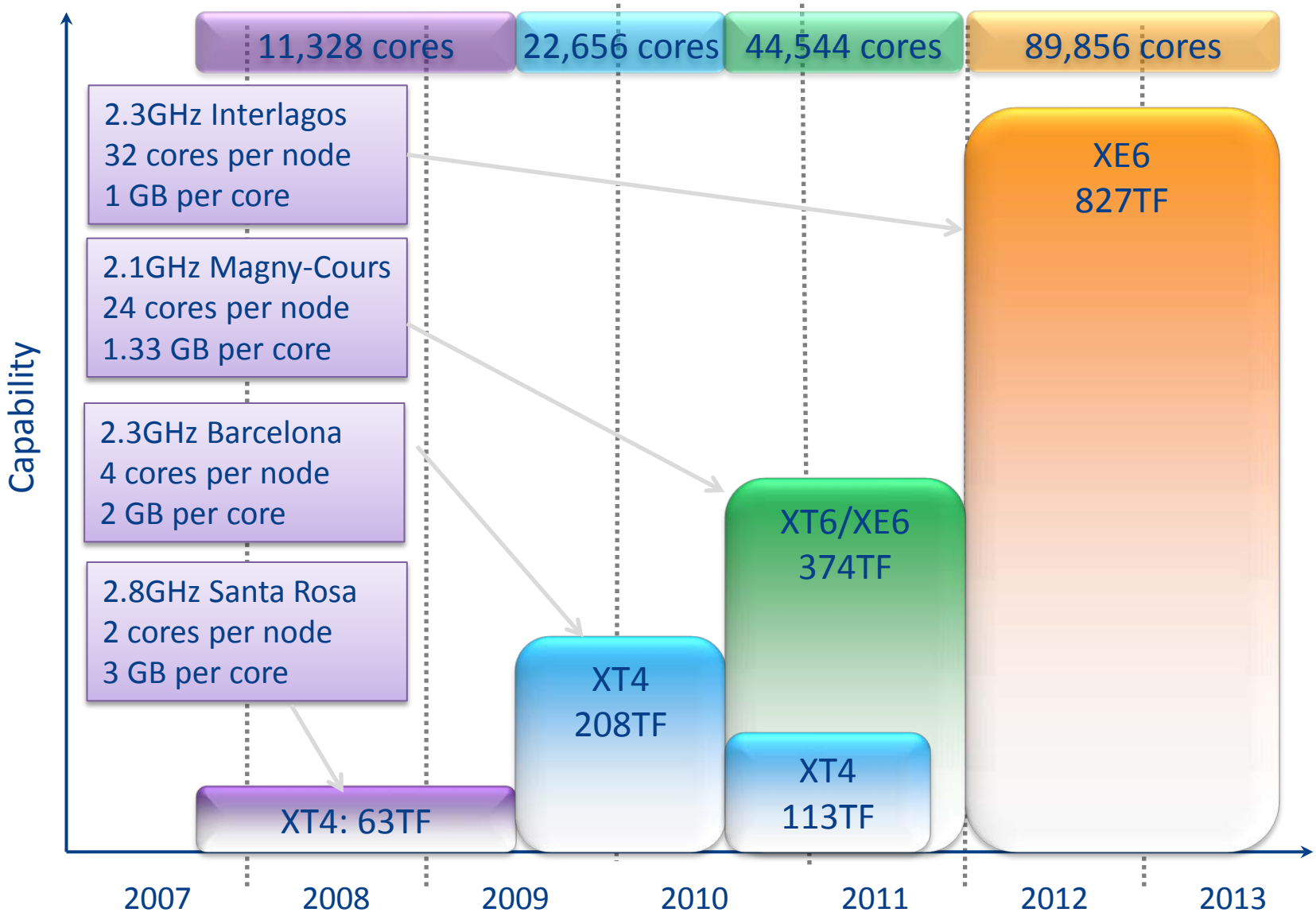
Overview

- Introduction to NAG
- General hardware trends
- Thoughts about Library design
- Conclusions & Discussion

Numerical Algorithms Group Ltd

- Founded 1970
  - Co-operative software project
  - Not-for-profit Company since 1976
- NAG Library released 1971
  - Currently at mark 24 with 1784 routines
- ~£8m financial turnover
- Strong links to academia
  - Sponsor PhD students, collaborative projects etc.
- Current business areas
  - Numerical and Statistical Libraries (finance, engineering, R&D, …)
  - Consulting: Code development, tuning, tailoring (finance, AMD, …)
  - HPC Services/Computational Science & Engineering (HECToR, CHPC, …)

Example: HECToR



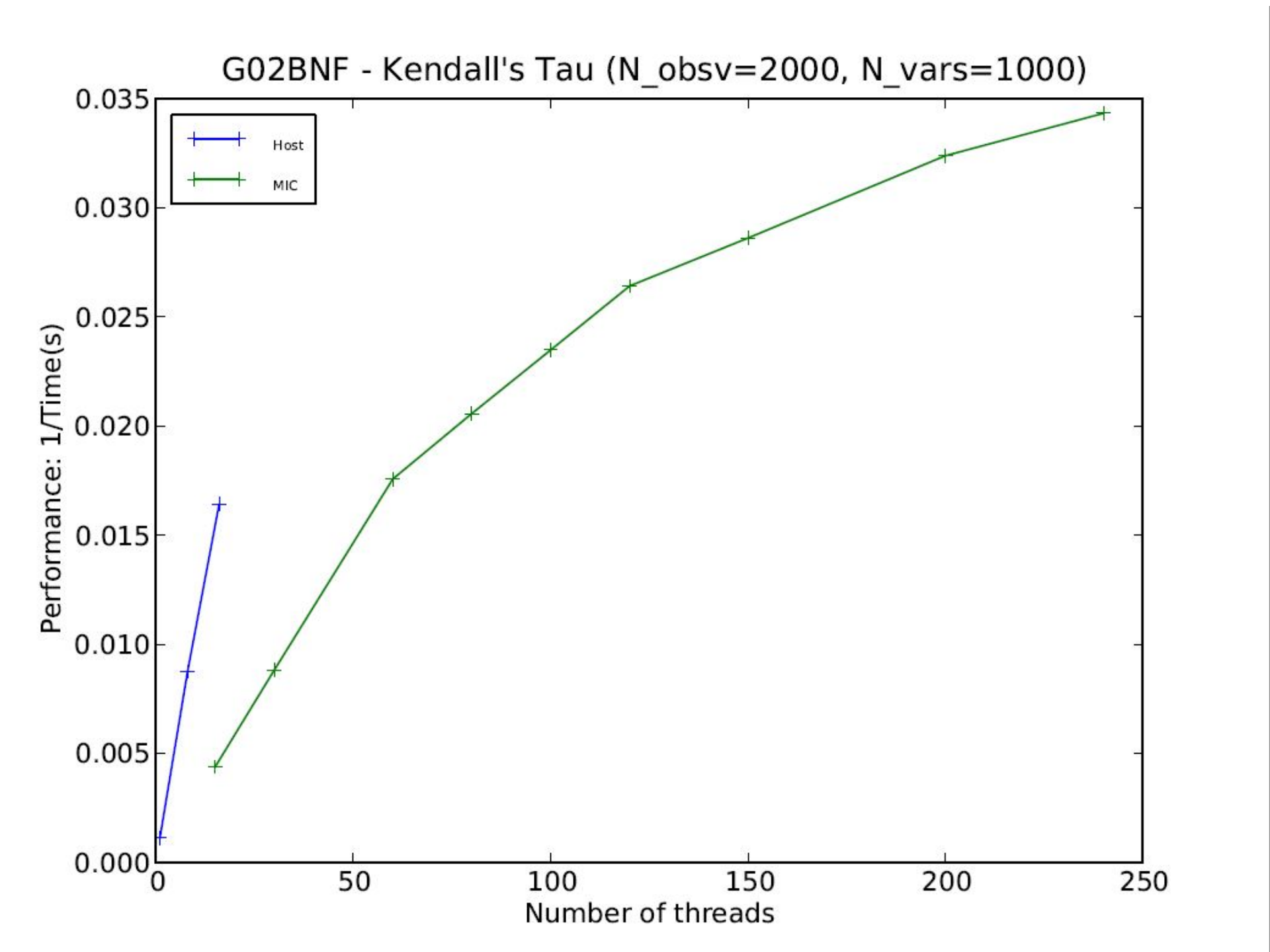
e.g. Xeon SNB

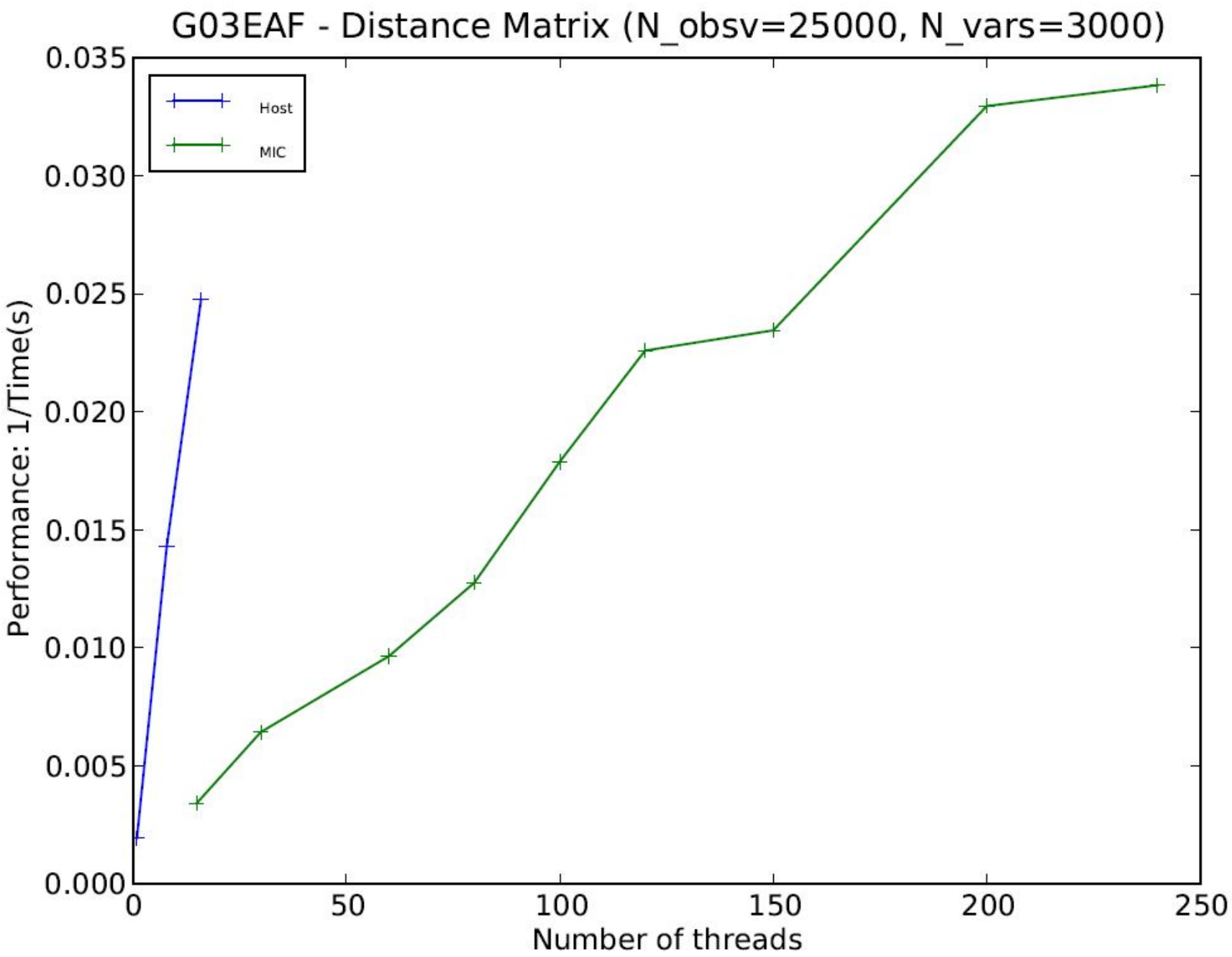
2.6 GHz processor 8 cores (FPU) per processor 8 DP FLOPs/cycle per FPU	Theoretical Peak Performance (GFLOPS)	Against Maximum
multi-threaded and vectorized	166.4	100.00%
multi-threaded & not vectorized	20.8	12.5%
serial and vectorized	20.8	12.5%
serial and not vectorized	2.6	1.6%

>**98%** of the possible FLOPS performance comes from parallel processing techniques

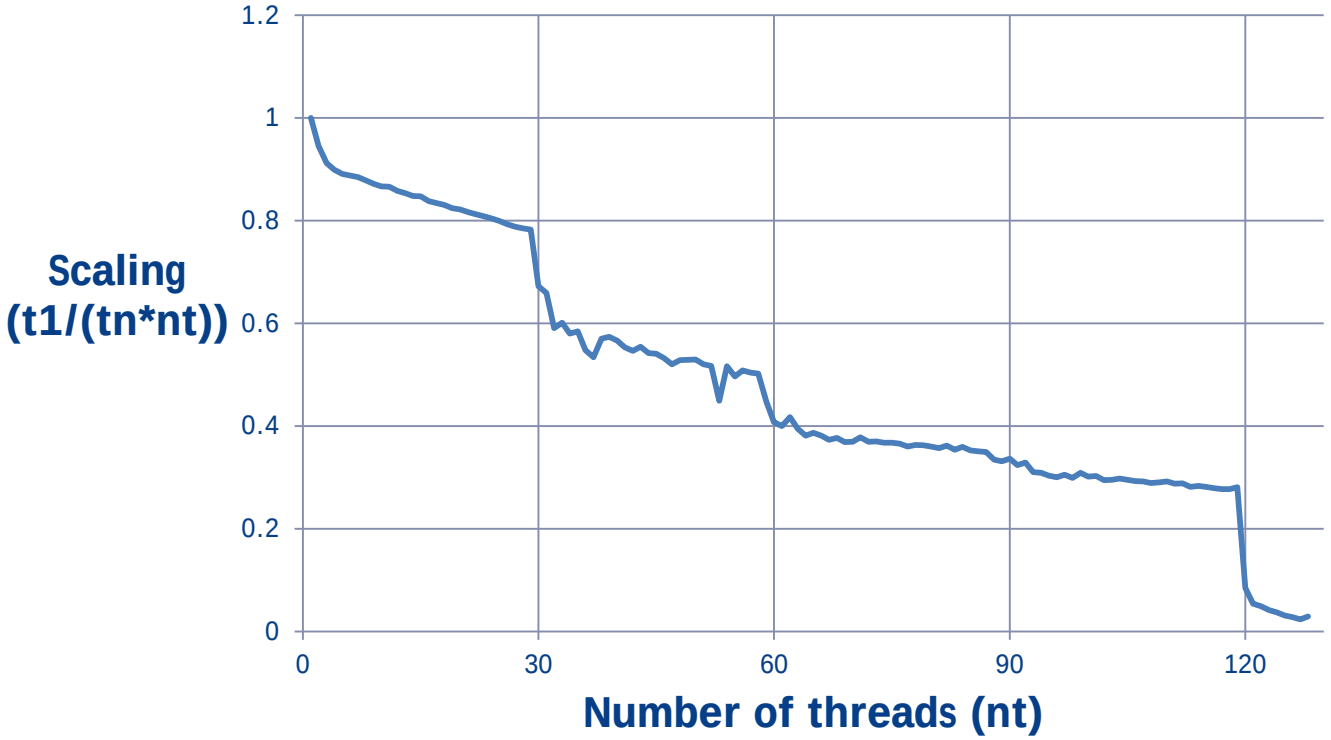
Accelerators

- Energy efficient
- Lots of cores – limited memory
  - E.g. Xeon Phi 7100p has 61 physical cores @ 1.238GHz, 244 virtual cores and 16GB memory
- Offload costs
- Unpredictable host environment
  - Need to tune for user's combination of CPUs and accelerators


G02BNF - Kendall's Tau (N_obsv=2000, N_vars=1000)
Host
MIC
Performance: 1/Time(s)
Number of threads
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250


G03EAF - Distance Matrix (N_obsv=25000, N_vars=3000)
Host
MIC
Performance: 1/Time(s)
Number of threads
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250

Kalman filter scaling example using MKL



- Parallelises over DGEMV and Householder reflections (MKL)
- Memory bound, hence drop-off in scaling at 29 threads

A workstation (or HPC node) might have …

	Peak DP GF	Clock GHz	DP FLOPS parallel	Total parallel
Opteron (Abu Dhabi)	160	2.5	8fpu x 8ops	64
Xeon (Sandy Bridge EP)	166	2.6	8fpu x 8ops	64
Xeon Phi (Knights Corner)	1,074	1.1	61c x 16ops	976
Tesla GPU (Kepler K20x)	1,312	0.7	14sm x 64c x 2ops	1,792
FirePro GPU (S10k Tahiti)	1,478	0.8	2gpu x 28cu x 4v x 8ops?	1,792

NB: excluding the whole FMA confusion

Memory

- Memory bandwidth has failed to increase in line with available flops
- Use of libraries inhibits
  - loop fusion
  - function inlining

  … leading to increased memory access cost
- See e.g. Built to Order BLAS http://ecee.colorado.edu/wpmu/btoblas/
- Increasing need to manage memory/data structures at the application level

Traditional Library Design

- Input:
  - Problem definition
  - Algorithmic options
- Output:
  - Problem result
  - Measure of accuracy
  - Feedback on solution process
- Ease of use
  - Sensible default values
  - Natural data structures
- Robustness
  - Never give the wrong answer or fail unexpectedly
- Modularity
  - Should be able to combine components in sensible ways

Issues with libraries

- Need to find and exploit more parallelism
  - Modular design not so good
  - Possible trade-off between mathematical rigour and performance
- Need to make better use of memory
  - Shouldn’t impose data structures on users
  - Should make better use of mixed precision algorithms
  - Need hooks to allow application to manage memory better
- Reproducibility of results

Reverse-communication interfaces

- Already used in the NAG Library
  - Useful for interacting with other software environments, e.g. calling Fortran from Excel
- Don’t pass whole problem description to routine, pass a piece at a time
- At intermediate stages, routine requests data from user
- User manages traversal of data structures ☺
- Much more complicated to use than conventional design ☹

Summary

- General hardware trends:
  - Available flops increasing but dependent on parallel programming
  - Overall memory and memory bandwidth increasing much more slowly
- Software always evolves more slowly than hardware
- Need to minimise memory access, and do as much work as possible per memory access
- Need to focus on algorithms – and implementations of algorithms – that parallelise (and vectorise) well
- Libraries are still worthwhile!