Unlocking the Power of OpenMP

Dr Stef Salvini
NAG Ltd
stef.salvini@nag.co.uk

Over 30 years of mathematical excellence
Acknowledgements

- Lawrence Mulholland
- Edward Smyth
- Themos Tsikas
- Anne E. Trefethen
- Jeremy Du Croz
- Robert Tong
- And too many others to mention here!
Opening Remarks

SMP Systems, Parallelism and OpenMP

Tuning OpenMP:
- General Points
- Nag Specific Case: Needs, Solutions and Case Studies

OpenMP and Hybrid Parallelism

Some Other Considerations and Conclusions
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP in Hybrid Parallelism
- Some Other Considerations and Conclusions
NAG and HPC

- **NAG Products**
  - Parallel Library Release 3 (MPI)
  - NAG SMP Library Release 2 (Open MP)
- **Collaborations with external agencies**
  - Vendors (e.g. ACML Library for AMD Opteron)
  - Research and academic institutions
  - Industrial, commercial and financial concerns
- **Consultancy activities**
Who Can Benefit from Parallelism?

- Anybody with large computationally intensive problems
  - Academic institutions
  - Industry (Aerospace, car, etc.)
  - Financial institutions (Forecasts, etc.)

- Increasingly commercial systems
  - Databases
  - On-Line Transactions
  - Data mining
  - Web Servers, etc.

They want solutions!
Some thoughts...

Hardware  ➔  Short Life Cycle

Scientific Software  ➔  Long Life Cycle

Is Software the Real Capital Investment?
The Changing World of HPC

- The “(ir)resistible” rise of PC Clusters
  - Price/performance (claims or substantial?)
  - Originally in-house built, now part of mainstream
  - Increasing penetration of the server market

- MPI as New Legacy
  - Is Parallelism crystallised into MPI?
  - Is anybody interested in other types of parallelism?

- Hybrid systems
  - The *de facto* standard for high-end systems
  - Do we need multi-level parallelism?
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP in Hybrid Parallelism
- Some Other Considerations and Conclusions

Over 30 years of mathematical excellence
Why SMPs?

- **Hardware**
  - Re-usable technology
  - Modular technology
  - Increasingly partitionable hardware
  - Reliable Technology (minimum downtime)

- **Commercial applications**
  - Databases
  - OLTP
  - Web Servers

- **Numerical and Scientific Applications**
  - Tremendous potential
SMP Model

CPU
Cache

CPU
Cache

CPU
Cache

CPU

Memory

Interconnect Subsystem

Single memory visible to all processors

Memory can be physically partitioned (NUMA systems)
Multi-threaded Parallelism (Parallelism-on-demand)

- Parallel execution
- Serial interfaces
- Details of parallelism are hidden outside the parallel region

Parallel Region

Spawn threads

Destroy threads

Serial execution

Multi-threading

Parallelism carried out in distinct parallel regions

Over 30 years of mathematical excellence

Stef Salvini

EWOMP 2003 - 22-26 September 2003
Data must migrate through the different levels of memory in a very coordinated fashion.

- Registers
- Primary Cache
- Secondary Cache
- Local Memory
- Global Memory
Memory and Data Transfer

- Memory Structure
  - Multiple-levels (increasing)
  - Caches invaluable, essential and difficult
- Some difficulties with data access
  - Single processor effects
    - Cache misses and thrashing
    - TLB misses
  - Multi-processor effects
    - False Sharing
    - Required synchronisations
- NUMA Systems
  - Data allocation and distribution
  - Page misses and migration
SMP Parallelism: Dynamic View of Data

Computation Stage 1

Data

Processor 1
Processor 2
Processor 3
Processor 4
From SMP to OpenMP

- OpenMP embodies SMP Mechanisms
- Concise notation
  - Not all parallel structures represented
- Simple
  - To implement: hence wide acceptance by vendors
  - to understand (?)
- Compiler Directives
  - Compile cleanly on serial systems
- However:
  - “Local” references only
  - Some system calls

Over 30 years of mathematical excellence
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP in Hybrid Parallelism
- Some Other Considerations and Conclusions
Ensuring Efficiency on Modern Systems

- Algorithm Must take into account
  - Parallelism
  - Data access (multi-level memory layout)

- Algorithms must have
  - Dynamic load balancing
  - Some strategy for serial or quasi-serial bottlenecks (beating Amdahl’s law?)
  - Parametrised for easy configuration & porting

- Should also algorithms take into account of
  - Multi-level memory (e.g. NUMA, clusters, etc)?
  - Contingent (history of the computation) data layout?
Levels of Parallelism

**Coarse Grain**
Application driven
More potential for parallelism
Closer to optimal performance
Less overheads
Design complexity
Implementation costs
Maintenance costs
Removed from “serial” world
“Non-local” data access problem
Non modular design
Expandable to non-SMP systems

**Fine Grain**
Close to “elementary” algorithms
Modular design
Direct path from “serial” world
Top-down refinement
Higher overheads
Less potential for parallelism
Serial or quasi-serial bottlenecks
Level of Parallelism

- Coarse/Fine trade-off dictated by
  - Nature of application
  - Technological feasibility
  - Availability of Components
  - Expertise and experience
  - Time scale and resources for development
  - Deadlines for results
The “reductionist approach”

- Postulate: Parallelise the basic computational kernels
  - BLAS (Basic Linear Algebra Subroutines)
    - Level-3 (Matrix-matrix product)
    - Level-2 (Matrix-vector product)
    - Level-1 (vector operations)
  - Basic FFTs
- Theorem: Anything built on them will have adequate parallelism
  - Proof by oral tradition
Level-3 BLAS

- $N^3$ Operations
- $N^2$ Data References
- Good data re-use (cache friendly)
- Good parallelisability
Level-2 and Level-1 BLAS

- $N^2$ and $N$ Operations
- $N^2$ and $N$ data References
- Poor data re-use
- Dubious parallelisability
- Problems if sequences of BLAS are applied to the same data space
DSYMV (Symm. Matrix Vector Product)

- Requires Synchronisations, etc
- Many vendors do not parallelise it!
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP in Hybrid Parallelism
- Some Other Considerations and Conclusions
NAG SMP Library

- **Our needs**
  - Fill the gap between Serial and Parallel codes
  - Allow maximum reuse of our products (NAG Library)
  - Best performance achievable (parallelism must make a difference!)

- **Our Approach**
  - Build the Library based on our NAG Library (serial)
  - Keep identical *serial* interfaces
  - Keep identical *functionality* and *numerics*
  - Hide all details of parallelism
  - Use parametrisable algorithms for easy porting
  - That dictates our level of granularity!
NAG SMP Library Release 2

- All the NAG Library Mark 19
  - Over 1200 user-callable components
- Parallelised numerical Routines in
  - Dense linear algebra
  - Sparse Linear Algebra
  - FFTs
  - Random-number generation
  - All other routines dependent on the above

Over 30 years of mathematical excellence
New at NAG SMP Library Release 3 (Soon)

- Sparse Technology
  - Direct Methods
  - Preconditioners for Iterative Solvers
  - Enhanced Iterative Solvers
  - Band Solvers
  - Eigensolution of Large Sparse Matrices

- Extended Linear Algebra Coverage

- Multi-Dimensional Quadrature
LU Factorisation: A “Serial” Algorithm
LU Factorisation: LAPACK Style

Active submatrix

Update the Trailing Submatrix (Level-3 BLAS)

Solve the triangular system (Level-3 BLAS)

Permute the Rows (Level-1 BLAS)

Factorise the pivot block (Level-2 BLAS)

L Factor

U Factor

Solve the triangular system (Level-3 BLAS)

Permute the Rows (Level-1 BLAS)

Factorise the pivot block (Level-2 BLAS)

L Factor

U Factor

Solve the triangular system (Level-3 BLAS)

Permute the Rows (Level-1 BLAS)

Factorise the pivot block (Level-2 BLAS)

L Factor

U Factor
About serial or quasi-serial bottlenecks

- Identify serial bottlenecks
- Identify memory access bottlenecks
- Remove the above or …
  - “Hide” them using a “look ahead” strategy
  - “Locally asynchronous” algorithms?

That is what we do

Over 30 years of mathematical excellence
About serial or quasi-serial bottlenecks

- Live with them …
- The reductionist approach
  - Further parallelisation of the bottlenecks
  - Perhaps nested parallelism?
- Use knowledge about the algorithms
  - Predecessor/successor
  - Task queues
- Use tools
  - Instrument the code to generate a stack of tasks
  - Profile and analyse previous runs
- “Hide” them using a “look ahead” strategy
  - “Locally asynchronous” algorithms
Look-ahead (Beating Amdahl’s Law?)

Parallelisation

Serial Execution

Parallel Execution

Parallel Execution with Look-Ahead

Over 30 years of mathematical excellence

Stef Salvini

EWOMP 2003 - 22-26 September 2003
LU Factorisation: LAPACK Style

Factorise the pivot block (Level-2 BLAS)

Permute the Rows (Level-1 BLAS)

Update the Trailing Submatrix (Level-3 BLAS)

Solve the triangular system (Level-3 BLAS)

L Factor

U Factor

Active submatrix
LU Factorisation: SMP Style (1)

On ALL processors:
- Permute the rows
- Solve the Triangular system
- Update

Active Submatrix
- Pivot Block, Factorise on 1
  After the update on 1 is done

U Factor
- Already Factorised
Apply all the permutations from the right in parallel.
LU Factorisation

Problem size (N) vs. Performance (Mflops) for different processor counts and systems:

- **Sun F15K – 1050 MHz**
- **Sun Perflib**
- **NAG SMP Release 2**

The chart illustrates the performance of different systems and processor counts for varying problem sizes. The x-axis represents problem sizes (N), and the y-axis represents performance (Mflops). The chart includes different processor counts, indicated by the legend, showing how performance scales with the number of processors.
QR Factorisation

Sun E6800 – 900 MHz

LAPACK
Sun Perflib
NAG SMP Release 2

Problem size (N)
Performance (Mflops)

Sun  E6800 – 900 MHz
NAG SMP Release 2
LAPACK
Sun Perflib
NEC SX4, LU Factorization

Problem size (n)

Performance (Mflops/sec)

NAG SMP Library

LAPACK

N. Procs

1
2
4
8
12
14

Stef Salvini
EWOMP 2003 - 22-26 September 2003
Over 30 years of mathematical excellence
LU Factorisation

Sun F15K – 1050 MHz

Execution Time (secs)

Problem Size (N)

Number of Processors

0.0
50.0
100.0
150.0
200.0
250.0
300.0
350.0
400.0
450.0
500.0
550.0
600.0
650.0
700.0
750.0
800.0

1 2 4 8 16 24 32 48

Over 30 years of mathematical excellence

Stef Salvini

EWOMP 2003 - 22-26 September 2003
Other cases

- **Tridiagonalisation**
  - Half operations through DSYMV
  - Half operations through rank-\(k\) symmetric update (Level-3 BLAS)
  - The gateway to symmetric eigenproblem

- **Full SVD (QR Algorithm) of a bidiagonal matrix**
  - Computational kernel: line rotations (Level-1 BLAS DROT)
  - Statistics, LLS problems, rank-deficiency, etc
Tridiagonalisation (Upper variant)

Problem size (N)

Performance (Mflops)

N. Procs

Sun F15K – 1050 MHz
Sun Perflib
NAG SMP Release 2

Over 30 years of mathematical excellence
Full SVD of Bidiagonal Matrix (QR Algor.)

Problem size (N)

Performance (Mflops)

N. Procs

1
2
4
8
16
24
32
48

Sun F15K – 1050 MHz
Sun Perflib
NAG SMP Release 2

Over 30 years of mathematical excellence
Full SVD of Bidiagonal Matrix (QR Algor.)

Sun F15K – 1050 MHz

Sun Perflib

NAG SMP Release 2

Execution Time (secs)

Problem Size (N)

N. Procs

Over 30 years of mathematical excellence

Stef Salvini

EWOMP 2003 - 22-26 September 2003
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP and Hybrid Parallelism
- Some Other Considerations and Conclusions
Clusters of SMPs

- **Future**
  - High-end
  - Medium-end
  - Low-end

- **Technology**
  - re-usable
  - upgradeable
  - Linux boxes?

**Hybrid (Mixed) Model?**

- NAG currently actively involved

Over 30 years of mathematical excellence
Some Considerations

- Enormous increase of CPU performance
- Less marked improvements to memory subsystems
- Using modular components

Relatively higher latency than currently
Hybrid Model Paradigm

Currently:
All processors the same (MPI, etc)

“flattening mountains” …
Hybrid Parallelism: Why?

- High-Latency systems
- Increased levels of memory
- Part-Serialisation of Message-Passing
- Increased Number of Processors Competing for Communication
Mixed Mode Parallelism: A Model’s Goals

- Maximize code re-use
  - E.g., retain message-passing main code architecture
  - Use existing SMP techniques and technology
- Allow some form of top-down refinement
  - Identify bottlenecks in isolation from rest of code and improve their efficiency
- Exploit a problem different levels of granularity
  - Coarse granularity: mapped onto message-passing
  - Fine granularity: mapped onto SMP
- ‘Hide’ communication costs (look ahead again)
- Reduce load unbalance
- Perhaps best with problems consisting of loosely coupled components

Over 30 years of mathematical excellence
SMP on Clusters

- OS-Level SSDMS
- Compiler-level
- “Architecture-aware” OPenMP
  - Explicit page allocation, etc
- Retracing HPF?
- Also, very much the topic of tomorrow’s panel
Contents

- Opening Remarks
- SMP Systems, Parallelism and OpenMP
- Tuning OpenMP:
  - General Points
  - Nag Specific Case: Needs, Solutions and Case Studies
- OpenMP in Hybrid Parallelism
- Some Other Considerations and Conclusions
Performance of Real Applications

- Memory bound
- MPI “faster” than OpenMP
  - Data “segregation”
    - Better access to memory
    - Limited cross-processor memory effects

We need
- Block algorithms
  - Better data locality
- User-specified prefetching

Over 30 years of mathematical excellence
Example: Very Sparse Large Problems

- Random matrix entries, almost diagonally dominant
  - Diagonal preconditioning
  - Virtually removes the effects of preconditioning from the performance analysis

- 4 Case studies
  - Neither matrix nor vectors fit in secondary cache
    - $N = 1000000, \text{NNZ} = 10000000$, random pattern
    - $N = 1000000, \text{NNZ} = 10000000$, random pattern within narrow band bandwidth = 200)
  - Matrix does not fit in secondary cache
    - $N = 153600, \text{NNZ} = 15000000$, random pattern
    - $N = 153600, \text{NNZ} = 15000000$, random pattern within narrow band bandwidth = 200)
Some Scalability Results

- TFQMR Random, n=1000000
- TFQMR Narrow band, n = 1000000
- Bi-CGSTAB (4) Random, n =153600
- Bi-CGSTAB (4) Narrow band, n = 153600

Number of Processors

Speed-up
Performance of Matrix-Vector Product

Relative CPU times

Case 1  Case 2  Case 3  Case 4

Arbitrary Units

Over 30 years of mathematical excellence
Some Food for Thought

- Algorithms producers
  - New algorithms required
    - Block algorithms
    - latency-tolerant
    - parallel-adaptive
    - dynamically load-balanceable

- Other aspects
  - Expertise
    - Dearth of SMP and mixed-mode parallel expertise
    - Increasing need
What do we need in OpenMP?

- More flexible synchronisations
  - “Level crossings” (some wait, one releases)?
  - Partial barriers, relationships of precedence?
- More flexible work sharing mechanisms
- Data allocation/distribution
  - Avoid HPF constructs
  - Is good page migration sufficient on NUMA?
- User-specified prefetching
  - Essential for performance
  - Portable API (or part of OpenMP)
  - Compiler writers rather against it
- Some message passing mechanism?
Multi-Level Parallelism: Do We Need it?

- **Multi-level applications**
  - Loosely coupled components
  - SMP Nested parallelism?

- **Heterogeneous applications**
  - Very difficult to map on OpenMP, currently
  - SMP nested parallelism?

- **Hardware requirements (Hybrid systems)**
  - Clusters of SMPs
Summary

- Tuning of Open MP
  - Difficult but feasible
    - Considerable gains
    - Gateway between serial and parallel worlds
  - Current algorithms may need revision
  - Good prefetching essential in the future

- Challenges ahead
  - Developing “look-ahead” strategies for numerical algorithms
  - Mapping existing numerical algorithm to future architectures (clusters of SMPs)
  - Developing new “multi-level” algorithms
Thank you

for

your attention

Stef Salvini
stef.salvini@nag.co.uk