



NAG[®]

The Numerical Algorithms Group

Combining mathematics and technology for
enhanced performance

Contribution to the Panel Discussion

Dr Stef Salvini

NAG Ltd

stef.salvini@nag.co.uk

Over 30 years of mathematical excellence

- **Grammatical Simplicity and Power**
 - Understandable API
 - Mostly, system independence
 - Easy portability
 - Modular approach
 - Gateway between serial and parallel worlds
 - Potential great performance
- **How to preserve it and increase its power?**

Question 2

- **Easy access to “non-local” data**
 - Data outside scope of module (currently Common Blocks only)
 - E.g. Client-server models
 - E.g. Event-driven situations
 - Inter-modular synchronisation (“long range” effects)
- **Task queues**
- **Relationships of precedence**
 - Potential conflict between resolutions of these at compilation or run time?

Question 2

▪ **Nested Parallelism**

- Right now, a bit of a mess (feels like an “add-on”)
- Much better facilities for this!

▪ **Better synchronisation**

- Limited scope and extent
- “Level crossing” locks (many wait, one frees)

Questions 1 and 3

- **Data “placement”**: what does it entail?
 - Data placement transformation modules?
 - “Memory of previous lives in thread”
 - Data distribution by stealth?
 - MPI is good at that: logical structure, disjoint data sets, etc
 - MPI is a logical and structured framework
 - GNHPF languages (GNHPF is Not HPF)?
- **Open MP should ignore any underlying architecture!**

- **Demand-side data economics (just-in-time data availability)**
 - Data available when required
 - OS or its agents to supervise all aspects of data transfer
- **Prefetching**
 - Data independence from placement
 - Multi-level (**temporal** not architectural sense)
 - Immediate use (e.g. load into Level 1 cache)
 - Intermediate urgency (e.g. load into Level 2 cache)
 - Long-term need (transfer to local memory)
 - Mechanisms for read/write contentions

▪ Prefetching (cont.)

- Data-centred, not mechanism- or architecture-centred
- Architecture independence
 - Relative not absolute latencies of relevance
 - Portable (probably)
 - Parametrisable (in line of principle)
 - Usable for wide range of architectures
 - Serial
 - SMP (including NUMA)
 - Clusters and Hybrid
- We need a prefetching portable API
 - Essential for future architectures (increasingly driven by commercial applications)
 - Essential for future-proofing software
 - Essential for true portability

- The **GRID!!** (Finally the word appears!)
 - As of now, its currency:
 - Data
 - Applications
 - Remote accessing of systems etc
 - Software NOT in its current scope

- **Needs a common conceptual development platform**
 - Move from “modules” to “components” (these include context and are modifiable by and for it)
 - Mechanism and architecture independence
 - Highlight tasks
 - Highlight need of data and their temporal relations (NOT their spatial relations)
- **At once portable anywhere (although not necessarily efficiently)**
 - Real pick ‘n use algorithmic base!