Summary: "Performance evaluation of load balancing algorithms for parallel single-phase iterative PDE solvers."

1 Introduction

Consider data-parallel iterative solvers for partial differential equations, discretised on irregular grids. Grid is provided. Problem is to partition the grid and allocate submeshes to processors.

Review various schemes in the literature for tackling this problem. Single out 6 schemes for detailed comparision: "$P \times Q$" algorithm, two more complicated heuristic schemes, and three "physical" optimisation techniques. Enumerate 8 criteria for comparing the schemes.

2 Data Mapping

Two basic strategies are to directly minimise a cost function dependent on communication and computational performance parameters of target machine, or to abstract the partitioning problem in a machine-independent way, then allocate the resulting partition. Possible objective functions for the former strategy are reviewed, and criteria for partitioning under the latter strategy are listed.

3 Communication Requirements for Parallel Iterative PDE Solvers

A PDE solver contains i) Local computation; ii) Local communication (edge swaps); iii) Global synchronisation (reduction operations).

4 Data Mapping Algorithms

Outline the 6 algorithms:

1. "$P \times Q$"
2. Recursive spectral bisection
3. "Geometry Graph Partitioning" algorithm
4. A Genetic algorithm
5. Simulated annealing
6. A Neural Network algorithm

3 "physical" schemes are expensive, and require the grid to be contracted before they can be applied.

5 Performance Evaluation

Apply all 6 schemes to a single problem—2D Poisson equation on an irregular domain. Discretised by finite element method onto mesh with 12,724 nodes. Choose machine parameters for "physical" optimisation schemes—nCube II.

Resulting mappings are compared for machine-independent qualities: number of interface (edge) nodes; connectivity of submeshes (valence of graph of submeshes?); maximum distances between connected submeshes, according to processor allocation (is this really machine independent?); nodes per submesh (load balancing); disconnectednes of submeshes.

Mappings compared for machine dependent qualities—run-times.

6 Summary of Results

7 Conclusions

Conclude that differences in qualities of schemes not sufficiently pronounced to justify use of more complex mapping schemes. The simplest $P \times Q$ heuristic seems adequate—and is very fast to apply.

Referees Comments (for Author and Editor)

Minor points:

- The authors refer to "compile-time mapping". It is unclear in what sense the data mapping could be "compile time". Are languages envisaged which could represent complex grids as compile-time constant expressions?
- Section 2, notation: Why D^h? (What is the superscript for?)
- The notation
$$\{D_i^h\}_{i=1}^P$$
is unfamiliar. Why not use a standard notation like $\{D_i^h \mid 1 \leq i \leq P\}$, or simply $\{D_i^h\}$?
- Section 5.1, para 3: "the total message latency is proportional to the submesh connectivity". Please explain.

I found the conclusion—that the most naive scheme works as well as the more complicated ones—appealing. I think it may be controversial.

Other Comments (Grammar, etc.)

- End of section 4.3:

  > "The GBA represents the partitioning graph [...] and the processors interconnection graph into a simpler space".

  Not grammatical.

- Section 5, para 2:

  > "[...] approximated by a linear system with *11,676 number of equations*"

  (my italics). Not grammatical.

- Section 5, para 3, sentence 3: "Also" misspelt.

- Section 5.1, para 4, sentence 2: "and" misspelt.

- Section 5.1, para 5, sentence 5:

  > "[...] the total sum [...]".

  Redundant use of "total".