

Chapter 3

Modelling

1 Design Aims

Designing and building
parallel programs. Foster

Metrics

A problem to solve
a candidate solution.
is it a solution?

We must have a set of criteria (metrics) with which to
measure the solution.

Rarely simple
execution time
memory requirements
implementation costs
maintenance costs
Scalability
disk space
resilience
portability
network requirements
usability
delivery date

How do we identify a
solution

Often overlooked

Performance
Modelling

A list of common
ones

2 Design
Decisions

Costs

Informed decisions on alternatives

what are the costs?

Preferably before they are incurred.

How do we identify a
solution

Estimation technique

Manpower estimates
are a different
problem

scalability

bottlenecks

resource requirements

Performance modelling

Mathematical models to answer these questions.

Choose between competing solutions

Identify problems before they arise

Needs **reliable** empirical data

How to make an
interpret
measurements

Performance
Modelling

3 Distributed v
Grid & Cloud

Little textbook knowledge on these problems. Data on specific problems, some of it in the literature.

Distributed

Most models of distributed/parallel computing assume a single administrative domain and a single computational system, at a single site.

Cluster or farm

Communication costs

Distributed

The data paths are short, high performance, reliable. (often special purpose). Cost (time and money) are low and bounded.

Grid

The data paths may be long, standard performance, suffer from contention and unreliable. Cost (time and money) are high and unbounded.

Cloud

The data paths are completely standard. Contention. Extra charge for moving data.

Reliability

Distributed

High, systems often have automatic failover in the case of processor failure. External network failure unlikely to impact on job.

Grid

The system is unreliable jobs fail returning no error. Your system must be resilient.

I know of no specific models for cloud calculations

Network failure may prevent job retrieval

Performance
Modelling

4 Performance Modelling

Performance Modelling

Specification may include

- hard numbers for some metrics
- require optimization for some
- ignore others altogether

We will look at some standard methods and highlight their shortcomings.

Look at some general principles so that you can develop suitable models for your own problems.

Manpower estimation outside the scope of this course.

5 Amdahl's Law

Maximum speedup

$$\text{Speedup} = \frac{\text{execution time on single processor}}{\text{execution time on } n \text{ processors}}$$

Best performance is limited by sequential component of programme

If sequential fraction is $1/s$, then maximum speedup is s . *Amdahl's Law*

But

true if incrementally parallelise a sequential algorithm.
false if a parallel algorithm replaces a serial one.

Have you parallelised the existing algorithm as much as possible – yes

Have you got the best possible parallel answer – no

Return to the limits of applicability to Amdahl later.

Best known theorem
on

Have I gone as far
as possible at this
stage?

Quantitative venner

Performance
Modelling

Limitations

6 Bridge



Build the
uprights before
the deck

Millau viaduct – world's tallest bridge
Finished in just three years

Because

The deck was built off site and slid into
position from the ends of the bridge



Performance
Modelling

7 Extrapolation from Observations

Observe and deduce

My favourite – an observation depends on no assumptions. It is true.

Extrapolation – model dependant. Your model must be correct or it will give misleading results.

Problem size N , number of processors P .

Measure for $N=90$ and $P=1$ and compare with $N=90$ and $P=10$.

The measured speed up is 9.1 compared with the perfect 10.

Consider three models which can be given a rational explanation.

.

Model independant

8 Models

One: The speed up is linear in the processor number apart from a constant which represents the time for start and finish.

$$T_p = 1 + N/P$$

Two: There is part of the programme which has no speed up and a part whose computational complexity goes like N^2 whose speed up is linear.

$$T_p = N + N^2/P$$

Three: Parallelise the computational intensive part, but the multi-processor introduces an overhead which goes like the square of the number of processors

$$T_p = N^2/P + 0.12P^2$$

$$T_1 = 90 + 90^2/1 = 8190$$

$$T_{10} = 90 + 90^2/10 = 900$$

$$\text{Speedup} = 8190/900 = 9.1$$

Constant overhead

Parallelise the computational intensive part

Every processor must communicate with every other one

Model 2

Performance
Modelling

9 Conclusions from Models

You shouldn't be surprised to learn

Model 1 speed up is 9.1

Model 3 speed up is 9.1

For model 1

Processor to 100 Complexity at 90	47.9
-----------------------------------	------

Processor at 10 Complexity at 500	9.8
-----------------------------------	-----

For model 2

Processor to 100 Complexity at 90	47.9
-----------------------------------	------

Processor at 10 Complexity at 500	9.8
-----------------------------------	-----

For model 3

Processor to 100 Complexity at 90	9.8
-----------------------------------	-----

Processor at 10 Complexity at 500	9.9
-----------------------------------	-----

Increase the complexity and all of them perform similarly.

Increase the number of processors and model 3 wins (in spite of the quadratic dependence on processor number)

Model 1 speed UP

p		2.0	3.0	4.0	5.0	10.0	100.0	1000.00
n								
1.0		1.3	1.5	1.6	1.7	1.8	2.0	2.0
5.0		1.7	2.3	2.7	3.0	4.0	5.7	6.0
10.0		1.8	2.5	3.1	3.7	5.5	10.0	10.9
20.0		1.9	2.7	3.5	4.2	7.0	17.5	20.6
30.0		1.9	2.8	3.6	4.4	7.8	23.8	30.1
40.0		2.0	2.9	3.7	4.6	8.2	29.3	39.4
50.0		2.0	2.9	3.8	4.6	8.5	34.0	48.6
60.0		2.0	2.9	3.8	4.7	8.7	38.1	57.5
70.0		2.0	2.9	3.8	4.7	8.9	41.8	66.4
80.0		2.0	2.9	3.9	4.8	9.0	45.0	75.0
90.0		2.0	2.9	3.9	4.8	9.1	47.9	83.5
100.0		2.0	2.9	3.9	4.8	9.2	50.5	91.8

11 Model 1
Results (i)

Model 1 Normalised speed UP

p		2.0	3.0	4.0	5.0	10.0	100.0	1000.00
n								
1.0		0.7	0.5	0.4	0.3	0.2	0.0	0.0
5.0		0.9	0.8	0.7	0.6	0.4	0.1	0.0
10.0		0.9	0.8	0.8	0.7	0.6	0.1	0.0
20.0		1.0	0.9	0.9	0.8	0.7	0.2	0.0
30.0		1.0	0.9	0.9	0.9	0.8	0.2	0.0
40.0		1.0	1.0	0.9	0.9	0.8	0.3	0.0
50.0		1.0	1.0	0.9	0.9	0.9	0.3	0.0
60.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
70.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
80.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
90.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
100.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1

Model 2 Normalised speed UP

p		2.0	3.0	4.0	5.0	10.0	100.0	1000.00
n								
1.0		0.7	0.5	0.4	0.3	0.2	0.0	0.0
5.0		0.9	0.8	0.7	0.6	0.4	0.1	0.0
10.0		0.9	0.8	0.8	0.7	0.6	0.1	0.0
20.0		1.0	0.9	0.9	0.8	0.7	0.2	0.0
30.0		1.0	0.9	0.9	0.9	0.8	0.2	0.0
40.0		1.0	1.0	0.9	0.9	0.8	0.3	0.0
50.0		1.0	1.0	0.9	0.9	0.9	0.3	0.0
60.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
70.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
80.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
90.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
100.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1

Identical in this region and at this precision

13 Model 3
Results (i)

Model 1 and 3 Normalised speed UP

p		2.0	3.0	4.0	5.0	10.0	100.0	1000.00
n								
1.0		0.7	0.5	0.4	0.3	0.2	0.0	0.0
5.0		0.9	0.8	0.7	0.6	0.4	0.1	0.0
10.0		0.9	0.8	0.8	0.7	0.6	0.1	0.0
20.0		1.0	0.9	0.9	0.8	0.7	0.2	0.0
30.0		1.0	0.9	0.9	0.9	0.8	0.2	0.0
40.0		1.0	1.0	0.9	0.9	0.8	0.3	0.0
50.0		1.0	1.0	0.9	0.9	0.9	0.3	0.0
60.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
70.0		1.0	1.0	1.0	0.9	0.9	0.4	0.1
80.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
90.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1
100.0		1.0	1.0	1.0	1.0	0.9	0.5	0.1

3 better

3 stays higher
longer drops
away sharper

p		2.0	3.0	4.0	5.0	10.0	100.0	1000.00
n								
1.0		0.6	0.3	0.1	0.1	0.0	0.0	0.0
5.0		1.0	0.9	0.8	0.6	0.2	0.0	0.0
10.0		1.0	1.0	0.9	0.9	0.5	0.0	0.0
20.0		1.0	1.0	1.0	1.0	0.8	0.0	0.0
30.0		1.0	1.0	1.0	1.0	0.9	0.0	0.0
40.0		1.0	1.0	1.0	1.0	0.9	0.0	0.0
50.0		1.0	1.0	1.0	1.0	1.0	0.0	0.0
60.0		1.0	1.0	1.0	1.0	1.0	0.0	0.0
70.0		1.0	1.0	1.0	1.0	1.0	0.0	0.0
80.0		1.0	1.0	1.0	1.0	1.0	0.1	0.0
90.0		1.0	1.0	1.0	1.0	1.0	0.1	0.0
100.0		1.0	1.0	1.0	1.0	1.0	0.1	0.0

3 worse

14 Conclusions

Superiority in one part of the processor complexity diagram, does not guarantee superiority everywhere.

A fall off shapes may differ

Measurements need to cover a range of values

and **preferably** cover the region of interest.

.

Resource Requirements

A skilful implementation may use fewer instructions to achieve an outcome.

A constant fraction –

$$t(\text{poor}) = k * t(\text{bad})$$

A faster machine will similarly run a constant fraction faster.

$$t(\text{fast}) = c * t(\text{slow})$$

The factors c and k make no difference to how the resource requirements for a given algorithm grow with size.

Resource Requirements

Before running (or even designing) a solution to a problem we would like to know if it is soluble (at reasonable cost).

Not too long; not too much memory.

For many things the resource requirements **scale** with the size of the input problem.

They get larger in a predictable way.

How long? Difficult to answer.

How much more when we double the size? Easier.

Given an algorithm

How long depends on

machine speed

coder skill

16 Asymptotic analysis

Asymptotic analysis

Count how many operations it takes to complete an algorithm.

Depends on computer architecture – but will normally be a constant percentage increase or decrease.

78% or 135%. Not relevant asking how the time increases as the problem size gets very large.

Depends on implementation of the algorithm.

Again all reasonable implementations differ by a constant factor.

Example:

How long does it take to sort a list of numbers?

If they don't they are not implementing the same algorithm

17 Sort

List length **N**

Read through the list and find smallest item.

read

compare

possible store

increment pointer.

4N operations.

Repeat to find 2nd, 3rd, largest item.

only need to search (N-1), (N-2) items

Need to do N times

$$\begin{aligned} 4[N + (N-1) + (N-2) + \dots + 1] &= 4 * N*(N+1)/2 \\ &= 2 * (N^2 + N) \end{aligned}$$

Ignore the 2 and the N (irrelevant for large N)

Polynomial $O(N^2)$

If they don't they
are not
implementing the
same algorithm

Performance
Modelling

18 Travelling Salesman

Shortest route between N towns.

2 towns ... 1 route AB

3 towns ... 2 routes ABC, ACB

4 towns ... 6 routes ABCD, ABDC, ACBD,
ACDB, ADBC, ADCB

Computationally $O(N!)$ Also known as hard

Clearly there is a constant to actually create the routes and calculate the length of each route.

Length increases like N , but can be ignored in comparison with the $N!$

If they don't they are not implementing the same algorithm

19 Asymptotic analysis

Tells us how cost varies when N and P are large.

Is your problem asymptotic

Is the model correct?

Textbooks often use asymptotic analysis in characterising parallel algorithms.

Parallel because we are looking to solve large problems

Algorithm requires Order($N \log N$) or Order(N) processors.

For a computer scientist $N > 1000$ is perfectly reasonable. For less than 1000 another term may dominate.

AA says $1000N(\log N)$ is better than $10N^2$ and so it is again for $N > 1000$.

Some models assume communication costs are small.

Grid computation the costs are high. Cloud data transfer costs are high and the networks are unreliable.

Our earlier models go to zero efficiency at large N and P

Does asymptotic analysis make sense?

20 Suitable models

Aims

Create a model

- explain existing observations
- predict unmeasured situations
- be just complex enough

Provide a method to calculate

$$\tau = f(N, P, U,)$$

τ is execution time. (Elapsed time - start of first processor, finish by the last.)

- N problem size
- P number of processors
- U number of tasks

Assumption

Processors are communicating, computing or idling.

Theories should be as simple as possible **and no simpler.**

Other models

- MPI (message passing)
- Data parallelism
- Shared Memory

Not suitable

21 Multi-computer

Multi-computer

A set of Von Neumann machines and an interconnection network.

First Draft of a report on the EDVAC
John von Neumann
Contract No. W-670-ORD-4926
University of Pensylvannia
June 30th 1945

A powerful simple model

Each computer executes its own programme

It may read/write local memory

It may send/receive to other nodes

Read/write is cheaper than send/receive *locality*

Send/receive only depends on message length.

Other possible architectures

MIMD SIMD

PRAM Parallel Random Access Machine
*note for this all communications are equally expensive. **Unsuitable** for grid computing.*

Locality an important property of parallel/distributed computing.

Performance
Modelling

22 Multi-computer

Locality data in the tasks local memory are close.

Channels are either network connections or may be some form of interprocess communication on the same processor.

Multi-computer

A set of Von Neumann machines and an interconnection network.

1. A parallel computation consists of one or more tasks, tasks execute concurrently. The number of tasks may vary during the execution.
2. A task is a sequential programme and local store. A set of *inputs and outputs*. Define interface to environment.
3. A task may computes; read; write; send/receive messages; create new tasks; terminate.
4. A send operation is asynchronous, a receive is synchronous.
5. Output/input pairs may be connected in *channels* which may be created and deleted.
6. Tasks are mapped onto physical processors. A task runs on one processor. A processor may run more than one task.

Channels aid location independence

Channels model dependency. One task needs another to complete before proceeding

Often create more tasks than processors.
Helps with scalability.
Helps to mask communication delays

A virtual Von N. machine

Read/write to local store

A channel is a message queue into which a sender may place a message and a receiver remove a message.

23 Modularity

Related objects run on one machine. Objects on different grid nodes should not need communication.

Modularity

Not required but helpful if a task corresponds to a module.

Modules develop independently and interact through well documented interfaces.

Good analogy between task/channel and OO programming. Channels are the equivalent of method calls.

OO programmes and programming languages provide a natural basis for distributed computing **BUT** method calls are cheap and WAN calls are very expensive.

OO is a good starting point, but need to be applied carefully.

24 Timing

Timing

Processor may be idle, computing or communicating at any time.

$$T_{\text{com}} \quad T_{\text{net}} \quad T_{\text{idle}}$$

Execution time is then defined in one of two ways.

Total time on an arbitrary processor.

$$T^j = T_{\text{com}}^j + T_{\text{net}}^j + T_{\text{idle}}^j$$

Average time on all processors time.

$$T = (T_{\text{com}} + T_{\text{net}} + T_{\text{idle}})/P$$

$$T = (\sum T_{\text{com}}^j + \sum T_{\text{net}}^j + \sum T_{\text{idle}}^j)/P$$

Sum runs over all processors.

25 Computation Time

Computation Time

Depends on problem size (measured in some suitable manner)

If the parallel algorithm requires some calculation to be repeated in every task, then increasing number of tasks or processors may change the total time.

Introducing multiple CPUs may cause unexpected changes in computation time. Multi CPUs may have more cache (and/or memory) hence faster access is possible. Or larger problems may need to page to disk, slower.

Memory may be a more effective speed up. Do you buy 2.6GHz + 4 GB or 3.2 GHz + 2 GB

Operation of the parallel algorithm may be more effective

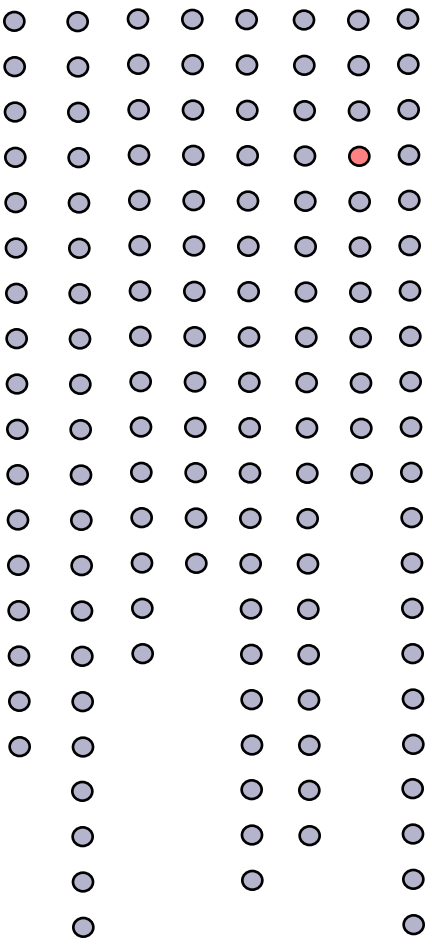
Accurate model may drive purchasing decisions.

26 Superlinear

Super Linear Speed Up

Expect the best we can get from adding N processors to a system is a factor N improvement.

Search problem..



$\tau_N = \tau_1 / N$ Linear

$\tau_N < \tau_1 / N$ Super Linear

An Error?

May serendipitously happen

Cache: N processors have more cache than 1 processor, if the problem on N processors fits into cache, but not on 1. Can see speed up. May also happen if 1 processor needs to page to disk.

Improvement in memory access greater than communication overhead.

Algorithm Operation some searches.

Search tree – move down from some set of initial values.

Single processor does each branch at a time. Multi-processor do the branches in parallel. If the average depth means on average less nodes are explored in the parallel case.

This does mean that the single processor algorithm is may not be optimal.

Performance
Modelling

Communication Time

Standard to distinguish.

Intraprocessor communication and interprocessor communication.

Intraprocessor: between processes in same CPU

Interprocessor: between processes in different CPUs.

Farms with good dedicated channels. *May* be little difference.

Context switching and copying data from one part of memory to another may be expensive

Grid communication: communications between different sites slower *unreliable*.

Grid introduces another level.

On chip v *Off chip*: New multi CPU chips are potentially faster. How do multi-chip architectures scale.

Multi CPU chips

Time is started up time (*constant*) plus time per word (*linear in message size*). Complications due to buffer size and buffer management may cause deviations.

Idle Time

Difficult to measure. Often caused by interactions between processes. Waiting for completion of other tasks. Contention for disk or network.

Possible to structure tasks to reduce idle time.

More than one task per processor. Overheads for task switching may be greater than idle time.

Explicitly interleave calculation and data fetch.

Scalability

Note we can either try to solve a problem faster.

Scalability at fixed problem size.

OR

Find running time for larger programmes. *Scalability with Scaled problem size.*

Often want to do both

Scalability of one does not imply scalability of the other.

29 Experimental tips

Tips

Decide what you are trying to determine before running. Have a model.

Make a number of measurements – fit, check against model. Extrapolate and repeat.

Measure as close to required configuration as possible.

If measurement on the required configuration is not possible, can we model a sub-system at full load

Repeated measurements are good. Averaging rarely is. Often outliers which give a false measurement.

Are we interested in mean – possible the maximum is more useful. Are the outliers due to some identifiable cause.

Full Time distribution has more information. Do we understand it.

Understand means predicting and confirming.

Problem Compute & communicate. Model communication by ignoring computation

Often seen in MSc thesis.

30 Experimental problems

Problems

Non-deterministic test problems may vary in size.
Random numbers – use reproducible ones to produce repeatable tests. Normalize on test size.

Inaccurate timing: if precision increase time to be measured. Find another measurement.

Startup and shutdown: Especially in a complex system.

Interference: Other programmes running.

Contention: limited resource – disk I/O, network bandwidth.

Standardise the environment.

Check you can predict normal environment from results in the standard environment.

PP collisions

Labs spend a lot of time setting up such environments.

Performance
Modelling

31 Other considerations

Save the state of the computation for subsequent restart.

Einstein said “Our description of nature should be as simple as possible, but no simpler”

A job which runs for many days will *checkpoint*. This may require significant amounts of temporary storage.

The checkpoint output may take a significant amount of time.

Data output – writing large amounts of data takes significant time *see checkpointing*

If the data structure is too big to hold in memory then paging will occur. Controlling data movement in and out of memory can lead to better performance.

Searching though data for interesting features may need much I/O storing and accessing data intelligently may improve performance.

A good epigram for computational models

Each processor searching a different disk