Software Performance Engineering

3C05 Advanced Software Engineering
04/11/2001
SEJPAL | PHULL

Agenda

- What is Software Performance Engineering (SPE)
- The goal of SPE
- When can SPE be used
- Why is SPE used
- Without SPE
- What it costs
- How SPE is Implemented & Measured
- Techniques for Improvement
- Future
- Summary

[hsejpal | tphull]@cs.ucl.ac.uk

What is Software Performance Engineering?

- Firstly what is performance?
  - "the degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy or memory usage" [IEEE 610.12]
- Next, what is SPE
  - Is the systematic process for planning and evaluating a new system's performance throughout the life cycle of its development. Its goals are to enhance the responsiveness and usability of systems while preserving quality

[hsejpal | tphull]@cs.ucl.ac.uk

The Goal of Software Performance Engineering

- To build predictable, adequate performance into systems by considering quantative behaviour from a system's requirements stage through to its maintenance and enhancement.
- Due to increasing system complexity, rapidly evolving software tools, performance engineering often represents the most challenging aspect of system building.

[hsejpal | tphull]@cs.ucl.ac.uk

When Can SPE be used?

- Software Performance Engineering is an integral part of the software development cycle – requirements to final transition
- Questions about performance metrics need to be answered – what and how to measure?
- Requires methods and procedures to extract meaningful information from a software system

[hsejpal | tphull]@cs.ucl.ac.uk

Why is SPE Used?

ADVANTAGES

- Improves hardware resource utilisation
- Performance engineering puts a different perspective on software design and development procedures. Implemented correctly, performance engineering reciprocates good, simple and efficient software.
- Extended thought to performance engineering leads to simple designs that are both correct and fast
  - Bell's Law: cheapest, fastest and most reliable components of a system are the ones that aren't there!

[hsejpal | tphull]@cs.ucl.ac.uk

Why Is SPE Used?

DISADVANTAGES

- Developers believe that methods needed are too demanding – prefer to adopt a ‘fix it later’ approach
- Its easier to detect performance problems after the system has been realised
- Common Myth
  - People believe that performance issues can be resolved by using faster hardware

[hsejpal | tphull]@cs.ucl.ac.uk

Without Software Performance Engineering

- “80% of all client/server projects have to be resigned because of performance issues, not because they didn’t meet the functional needs of the users” [INFO]
- Costs to companies such as
  - Business failures
  - Lost income
  - Reduced competitiveness
  - Damaged customer relations
  - Additional project resources

[hsejpal | tphull]@cs.ucl.ac.uk

What does Software Performance Engineering Cost

- Lucent Technologies has reported that the cost of SPE for performance-critical projects is about 2% - 3% of the total project budget
- Bank One Statistic
  - SPE Cost $147,000
  - Annual Saving $1,300,000

[hsejpal | tphull]@cs.ucl.ac.uk

Implementing Software Performance Engineering

- Efficient s/w allows users to use their h/w more efficiently
- Should be implemented and addressed at all stages in s/w development as an integral part of the development process
- SPE is a notoriously difficult and thankless task

"Problem prevention is a thankless act, but a problem resolution is a hero maker" [PENG]

- Various approaches to implementing SPE
- Area in s/w development that is widely overlooked

[hsejpal | tphull]@cs.ucl.ac.uk

Levels of SPE Improvement

System Structure

the most crucial level - dictates architectural performance

Modular Structure

defines performance of s/w data structures

Algorithm Implementation

defines performance of underlying algorithmic properties

Translation to Machine Code

defines compilation of program and machine code generation

Hardware Improvements

the way in which s/w performance is affected by system h/w

[hsejpal | tphull]@cs.ucl.ac.uk

SPE Approaches

Two Main Approaches to SPE

CURE

Build system and monitor under various workloads and conditions

PREVENTION

Model system quantitatively in order to analyse how system will perform

Is Prevention Better Than Cure?

[hsejpal | tphull]@cs.ucl.ac.uk

CURE

Build S/W System, Monitor Performance, Improve Performance

- ADVANTAGES
  - Exact measurements obtained
  - Easy to deploy
  - Fits well into an incremental development process
- DISADVANTAGES
  - Requires a working system
  - Impractical when developing time/safety critical system
  - Major performance deficiencies that require structural redesign will be extremely hard to reconfigure
  - Can result in long run expense - if s/w does not meet performance requirements

[hsejpal | tphull]@cs.ucl.ac.uk

PREVENTION

Model system, Monitor Performance, Evaluate / Improve Design

- ADVANTAGES
  - Problems predicted and addressed at earlier stage in s/w development process
  - Allows testing the impossible
  - Model unexpected scenarios to measure performance
  - Simulate scenarios that could damage the real world
  - Long term time and budget benefits
- DISADVANTAGES
  - Risk oversimplification
  - Cannot model everything - scenarios often missed out
  - Initially expensive

[hsejpal | tphull]@cs.ucl.ac.uk

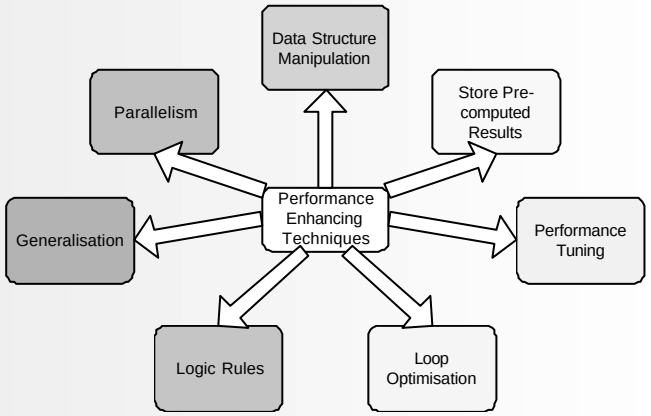
Measuring S/W Performance

- Before committing to a given design, calculate in rough terms the achievable s/w performance
- Requires use of s/w metrics to estimate / measure and analyse performance under various conditions
- How do we go about measuring s/w performance? - Instrumentation
  - Identify expensive operations by allocating cost in terms of system resources to these operations within the s/w
  - Cost is measured in terms of s/w metrics e.g.
    - % runtime / operation
    - Processed transactions / second
    - % of events causing failure
  - Use these metrics to accept or reject the system, dependant on client requirements, competition, expected workload etc.

[hsejpal | tphull]@cs.ucl.ac.uk

Performance Enhancing Techniques

- Once expensive operations are identified, we need to use performance enhancing techniques to optimise s/w performance
- Various techniques exist (formal / informal)



[hsejpal | tphull]@cs.ucl.ac.uk

Data Structure Manipulation

Augmentation
Often beneficial to augment data structures with redundant information to allow faster access to it

E.g. Accessing a file by line number - it is useful to build array whose elements point to the beginning of each line

Reduction / Elimination
Eliminate redundancies in sparse data structures to allow improved performance when accessing the data

E.g.Sparse data structures can result in wasted resources when applying algorithms to the data structure

[hsejpal | tphull]@cs.ucl.ac.uk

Store Pre-computed Results

- If a module / section of code computes a particular calculation several times, it would be efficient in terms of performance if the result of this calculation is stored as a partial result
- A form of caching - rather than computing an operation each time to obtain the result, the result is stored allowing fast access to it
- A form of Lazy Evaluation - only evaluating a result when it is needed

[hsejpal | tphull]@cs.ucl.ac.uk

Performance Tuning

- Minimising big-O complexity of the program - use of asymptotically efficient algorithms
- Avoid expensive algorithms where possible i.e. use $n\log n$ rather than n^2 complexity when choosing an algorithm
- Requires good knowledge of algorithms and their performance in terms of complexity
- Remember: Time-cost analysis should be machine independent (same implementations on different machines introduces a constant factor)
- Choosing suitable efficient algorithms requires knowledge of algorithm domain, frequency of use, best and worst case evaluations … see 2B12 Denise Gorse (Analysis of Algorithms)

[hsejpal | tphull]@cs.ucl.ac.uk

Loop Optimisation

- Effective coding of loops is a technique that can quickly improve s/w performance
- E.g.

```
for(int i=0; i<=10; ++i) {
    int x=10;
    System.out.println(x+i);
}
```

- The performance of the ‘for’ loop can be improved by initialising x outside the scope of the loop
- A simple and quick way of improving s/w performance

[hsejpal | tphull]@cs.ucl.ac.uk

Logic Rule Optimisation

- Optimising the evaluation of logical functions
- E.g.
  - `f(a) + g(b) > min`
  - Assume we know `f(a)` and `g(b)` are non-negative values
  - Improve long run performance by evaluating if `f(a) > min`
  - If `f(a) > min` then we know the logical function will be `> min`
  - If `f(a) <= min` then we evaluate `f(a) + g(b) > min`
  - Overhead of checking - so need to carefully deduce the domain of the function
- Conjunctive and Disjunctive logical functions can be ordered in code so to evaluate the cheapest operations first. This then allows us to try and 'short circuit' the the logical evaluation as soon as possible
- Basically, try and reduce the logical computation that the h/w needs to engage in through SPE techniques

[hsejpal | tphull]@cs.ucl.ac.uk

Generalisation

"the inventor's paradox"

- If is often cheaper in the long run to optimise a general solution rather than optimising a specialised solution
- E.g. sorting an array of 4999 elements
  - this can be optimised by writing a specialised routine to perform this task
    - `for (int i=0; i<4999; i++) { … }`
  - OR …
  - write a generalised routine that can sort an array of any size
    - `for (int i=0; i<arraySize; i++) { … }`
- Optimising a generalised routine is more complex, however in the long run this optimisation will pay dividends - it is easier than optimising many specialised routines

[hsejpal | tphull]@cs.ucl.ac.uk

Parallelism

- Improve s/w performance by effectively using concepts of concurrent programming
- Can lead to effective use of system processor(s) to improve s/w performance and enhance speed of operations
  - E.g. speed up system response by off-loading time consuming tasks to separate processes
  - E.g. Improve system throughput performance by using multiple processes to manage communication and device latencies
- However, requires complex modelling techniques to achieve concurrent design

[hsejpal | tphull]@cs.ucl.ac.uk

Performance Enhancing Techniques - Issues

- The techniques mentioned can be used to optimise s/w performance when implementing either of the two approaches:
  - CURE - *to improve the built s/w performance*
  - PREVENTION - *during design and modelling*
- Two major concepts and considerations when developing performance efficient s/w will be considered next:
  - Bottlenecks
  - Premature Optimisation

[hsejpal | tphull]@cs.ucl.ac.uk

Bottlenecks

- Some code is executed more frequently than other parts of code
- These frequently accessed areas are potential bottlenecks in the system
- Gene Amdahl's Law states,

  Speeding up code inside a bottleneck has a bigger impact on s/w performance than speeding up code outside the bottleneck
- Therefore, often, the frequency of code access is more important than the performance of a section of code
- 90/10 Rule: 90% of runtime is usually spent in 10% of code
- Amdahl - if a part of code is responsible for a fraction `f` of runtime, and can be sped up by a factor `s` - the overall speed will be:

  ```
  ((1-f) + f) / s
  ```
- Bottlenecks are hard to detect and are generally found within system provided code.

[hsejpal | tphull]@cs.ucl.ac.uk

Premature Optimisation

"the root of all evil" [Knuth]

- After the design of a performance aware system, you should not think about efficiency until testing is complete
- REASONS:
  - Optimising code often makes the code more complex hence the code is harder to debug
  - Very likely to spend times making areas of code performance efficient where it does not really matter
    - E.g. Improving a function from 10% of runtime to 6% of runtime is worthwhile, however, improving a function that is 0.001% of runtime to 0.0006% may not be as effective
    - If you optimise at this level, there may be thousands of operations that can optimised
  - Easy to get carried away and deferred from the s/w requirements
  - The design should incorporate s/w performance considerations, hence the architecture of the built system should be efficient

[hsejpal | tphull]@cs.ucl.ac.uk

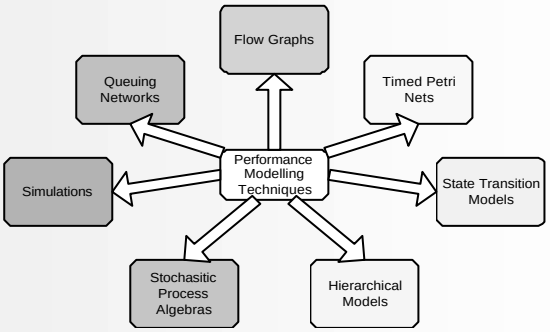
Prevention Better Than Cure...

- We have seen various techniques to improve s/w performance - these considerations should be taken into account when thinking of s/w design and implementation
- We have also seen the two main approaches of s/w development:
  - CURE
  - PREVENTION
- Now we will take a closer look at PREVENTION and specifically modelling as a technique to improve the s/w of systems

[hsejpal | tphull]@cs.ucl.ac.uk

Modelling

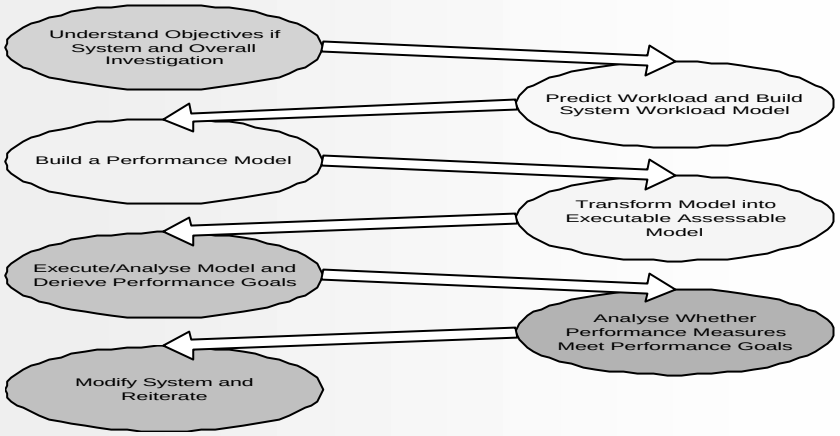
- The modelling approach is a way of representing the performance of an imaginary or incomplete system (see 2B12 Soren Sorenson - Systems Modelling)
- A widely studied and researched area of computing. Various modelling techniques based around mathematical and theoretical analysis have been developed to aid SPE
- Various techniques include the following



[hsejpal | tphull]@cs.ucl.ac.uk

Steps of Performance Modelling

Flow Diagram below shows the steps taken in order to deploy modelling as part of SPE



[hsejpal | tphull]@cs.ucl.ac.uk

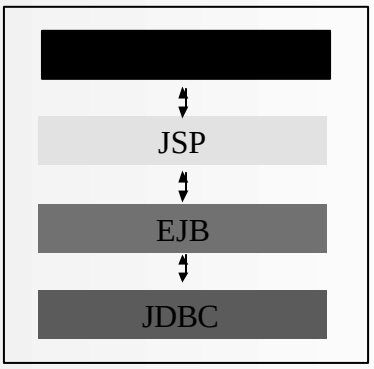
Stochastic Process Algebras

- Proposed as a tool for performance & dependency modelling
- Allows ability to model a system as the interaction of its subsystems
- So what exactly is it?
  - An extension of process algebras such as FSP where the time of an occurrence of actions is determined by random variables

[hsejpal | tphull]@cs.ucl.ac.uk

Application of Stochastic Process Algebras

- Consider the following 4-tier architecture



[hsejpal | tphull]@cs.ucl.ac.uk

Application of SPA's Continued

- Using Stochastic Process Algebra's we are able to model the system interactions before it is built
  - For example querying the database
  - Use SPA to model the architecture as individual components – done by modelling the system with a normal Process Algebra then mapping it onto SPA
  - Since the modelling takes place before the system is built time and resources are saved on prototyping and testing

[hsejpal | tphull]@cs.ucl.ac.uk

Future Trends in SPE

- A well understood formalism for performance engineering modelling - probably based on OOAD
- Embed performance engineering into a s/w development methodology
- Integrate performance engineering tools with current design and development tools
- Embed performance modelling as part of performance engineering and testing frameworks
- Automated performance engineering procedures
- Considerable research is still required into this subject

[hsejpal | tphull]@cs.ucl.ac.uk

Research Directions ...

- Integration of techniques and automated tools for performance engineering
- Formalise the procedure for performance engineering - integrate with current formalisms
- Embed performance engineering as part of new s/w development processes as an integral past of the process
- Current work comprises research into mappings from UML to Layered Queuing Networks and Stochastic Process Algebra Models

[hsejpal | tphull]@cs.ucl.ac.uk

Summary

- When and why SPE is used.
- Different Approaches
- Implementation of SPE
- CURE vs. PREVENTION
- Performance Enhancing Techniques
- Steps of Performance Modelling
  - An example: Stochastic Process Algebras
- Future & Research Directions

[hsejpal | tphull]@cs.ucl.ac.uk

References

- Text References
  - SMITH[1990] Performance Engineering of Software Systems Addison-Wesley
  - SOMMERVILLE[2001] Software Engineering, 6th Edition Addison-Wesley ISBN 0-201-39815-X
- Internet URL References
  - www.cs.mu.oz.au/252/s9_perf.tty
  - www.cs.utexas.edu/users/software/
  - www.peak.cs.hut.fi/research/13.html
  - www.perfeng.com/
  - www.softwaresystems.org/future.html
  - www.cs.ucl.ac.uk/staff/o.gotel/
- Professional Bodies
  - BCS Performance Engineering - www.cee.hw.ac.uk/~pjbk/perfeng/ Join electronic mailing list, send email to: uk-performance-request@cee.hw.ac.uk

[hsejpal | tphull]@cs.ucl.ac.uk