Contents

2.3 Compatibility

A solution to the concurrency problem must be compatible with existing programs and software development processes. Unfortunately, the changes needed to support concurrent execution are not always confined to the performance critical regions of the program. This section explores how the compatibility criterion restricts the design space of concurrent programming solutions.

The sole reason for writing a concurrent program is to obtain speed-up from the performance-critical regions of a program that can benefit from parallel execution. These regions are only a small part of most programs and a solution to the concurrency problem should only apply to these regions.

The main contribution of this section is the recognition that the potential benefits of concurrent execution are rarely compelling enough to justify disrupting existing software development processes or completely re-writing existing programs. This section focuses on defining the scope of possible solutions to the concurrency problem that are compatible with existing software.

2.3.1 Disruptive changes to existing software

The benefit of exploiting concurrency must exceed the costs associated with implementing it.

Section 2.3.3 explains why a worthwhile concurrent programming solution must improve total software development productivity.

Research into concurrent programming tends to focus on obtaining speed-up from the concurrent execution of regions of a program that benefit from it while giving little consideration to the impact on those regions of a program that do not. To be compatible with existing software a concurrent programming solution must only affect the regions of a program that benefit from concurrent execution.

To stand a realistic chance of adoption a concurrent programming solution should be compatible with existing software, libraries, operating systems, development tools and hardware.

Parallel programming is a mechanism for reducing the elapsed execution time of a program when the task dependencies are known, whereas concurrent programming addresses the cases when task dependencies cannot be known until the program is executed. Regions that benefit from parallel execution can occur

in the same application program as those that benefit from concurrent execution, so a concurrent programming solution should be compatible with a parallel programming solution.

During the unit testing phase of application development it must be possible to reproduce a problem for debugging purposes, during the acceptance testing phase it must be possible to stimulate all possible program behaviours and in production it must be possible to capture a program's behaviour so that errors can be reproduced. To be compatible with existing software a concurrent application must exhibit reproducible behaviour, so that it can be integrated into existing testing methodologies.

Thus, a concurrent programming solution must be locally applicable, compatible with existing software and development processes, compatible with a parallel programming solution and compatible with existing testing methodologies.

2.3.2 Compatibility with existing software

A concurrent programming methodology should be applicable locally and it should not be necessary to structure a program around the requirements of those regions that it is beneficial to execute concurrently. We found that, by focusing on the shared state interface and developing concurrent applications, rather than concurrent systems, we were able to restrict the locality of program changes to those routines that benefit most from concurrent execution.

A concurrent programming solution should be implemented in software, without requiring changes to the compiler, the operating system or the software development tool chain. We developed a concurrent programming solution in C++. The use of a conventional imperative language, compiler and development tool chain minimises the impact on existing programming methodologies.

A concurrent programming solution should be compatible with a parallel programming solution. We focus on providing compatibility with the Threading Building Blocks library [Int09]. Threading Building Blocks is an integrated parallel programming solution for Chip Multi-Processors. Our solution allows Threading Building Blocks to schedule both tasks that are known to be independent and tasks that may contain conflicting memory operations.

A concurrent execution environment should ensure reproducible application behaviour. We focus on using time stamps to ensure the correctness of concurrent execution. Time stamps can be used to ensure reproducible behaviour and to

determine the relationship between tasks during the problem solving process.

2.3.3 Making concurrent programs easier to write

The goal of research into concurrent programming is to make it easier to create scalable concurrent programs. To achieve this goal, the benefit from the reduction in the execution time of a concurrent program, relative to an equivalent serial program, must exceed the total cost associated with making that program execute concurrently.

A technique that makes program coding easier might make a program more difficult to debug offsetting any programmer productivity gains. Any proposal to make programming easier should improve productivity when amortised over the entire development process including: program design, coding, debugging, testing, operation and maintenance. The benefits of a new programming technique must also exceed the costs associated with learning it and the cost of rectifying mistakes made when it is applied incorrectly.

Regions of many types of application may benefit from concurrent execution, so the challenge is to integrate techniques to support concurrency into existing programming environments in such a way that utilising concurrency in those regions is worthwhile.

Bibliography

[Int09] Intel. Intel Threading Building Blocks: Programming for Current and Future Multicore Platforms. *IEEE/ACM International Symposium on Code Generation and Optimization*, July 2009.