Consulting Fire
Engineers
34 Satara Crescent
Khandallah
Wellington 6035

# High Performance Computing as a Resource for Fire Engineering Design

Prepared by: T.G. O'Brien

For: New Zealand eScience Infrastructure (NeSI)

Dated: 26 April 2016

Revision: Release 01

DDI: 001164 (0)4 479 3963 Mobile: 001164 (0)27 6413111 fe.aquacoustics.biz
FireNZE is a trading division of Aquacoustics Limited

**Table of Contents**


**1 Executive Summary ..... 4**

**2 Introduction ..... 5**
2.1 Note of Appreciation ..... 6

**3 Purpose ..... 7**

**4 Discussion - FDS Processing Speed ..... 8**
4.1 Amdahl's Law ..... 8
4.2 MPI and OMP ..... 8
4.3 Hyper-Threading ..... 9
4.4 Other Run-Time Improvement Strategies ..... 9
4.4.1 CPU Speed ..... 9
4.4.2 Turbo and Over-clocking ..... 9
4.4.3 CPU Architecture ..... 10
4.4.4 Network ..... 10
4.4.5 Processing Overheads ..... 10
4.5 Computational Domain Size ..... 10
4.6 Grid Resolution ..... 11
4.7 FDS Model Parameters ..... 12

**5 Project Methodology ..... 13**

**6 Computational Platforms ..... 14**
6.1 HPC Pan Cluster ..... 14
6.2 Beowulf Cluster ..... 14
6.3 Windows Workstation ..... 14

**7 Project FDS Model ..... 16**
7.1 FDS Version ..... 16
7.2 Geometry ..... 16
7.3 Computational Domain ..... 17
7.4 Meshes ..... 18
7.5 Design Fire Specification ..... 18
7.6 FDS Parameters ..... 19
7.7 Model Output ..... 19

**8 Project Test Cases ..... 21**

**9 Project Implementation ..... 23**
9.1 Post Simulation Processing ..... 23

**10 Results and Analysis ..... 25**
10.1 Single Mesh with OMP ..... 25
10.1.1 Processor Speed ..... 25
10.1.2 OMP Parallelization ..... 26

10.1.3 Run-Time and Computational Domain Size ..... 26
10.2 MPI (Multi-Mesh) ..... 28
10.2.1 Processor Speed ..... 28
10.2.2 MPI Parallelization ..... 28
10.3 MPI and OMP ..... 32
10.3.1 Platform Performance ..... 32
10.4 Run-Time Improvement ..... 33
10.5 Economy ..... 37
10.5.1 Cost ..... 37
10.5.2 Efficiency ..... 37
10.6 Variation in MPI Processing Time ..... 38
10.7 Variation in FDS Output ..... 40
10.7.1 FDS Verification ..... 40
10.7.2 Findings ..... 40
10.7.3 Discussion ..... 41

**11 Conclusions ..... 42**
11.1 Commercial HPC ..... 42
11.2 Parallelization ..... 42
11.3 Computational Domain ..... 43
11.4 Run-Time Variability ..... 43
11.5 FDS Model Output Variability ..... 43

**12 References ..... 44**

**Appendix A FDS Test Model ..... 46**

**Appendix B SLURM Script ..... 48**

**Appendix C BASH Script ..... 49**

**Appendix D Comparative MPI and OMP Graphics ..... 50**

**Appendix E FDS DEVICE Activation Times ..... 56**

# 1 Executive Summary

Fire modelling using Computational Fluid Dynamics (CFD) can present significant run-time challenges for real world fire engineering problems.

This project evaluates the run-time of Fire Dynamics Simulator (FDS) using currently available commercial High Performance Computer (HPC) resources in comparison with the more modest computer platforms usually available to fire engineers.

While there have been numerous studies of FDS performance and there is a significant body of literature on parallel processing strategies, both software and hardware have continued to evolve over time. A new study is therefore warranted.

This report also considers aspects of CFD run-time optimization through model refinements and parallel processing strategies to reduce model run-time. Many of the conclusions reached on these issues simply validate advice contained within the FDS User's Guide and Technical Manual.

The report concludes that commercial HPC facilities provide a viable resource for CFD, particularly when a number of moderate-to-large models must be run concurrently to meet project timelines. However modest computational resources remain a useful adjunct to commercial HPC for model development and for fast processing of a limited number of fire scenarios.

This report includes a comprehensive description of the simulation and the project methodology to allow other users to replicate the results on other hardware and software platforms for comparative purposes. Readers interested in the analysis and conclusions should refer directly to Sections 10 and 11.

## 2 Introduction

Computational Fluid Dynamics (CFD) can be a useful design tool for certain classes of fire engineering problem. The CFD program of industry choice is Fire Dynamics Simulator (FDS) by NIST[1] because the program is available for free, it has a wide user base, it is well-supported and documented, it runs relatively quickly, and has been subjected to extensive validation[6,18].

A problem that arises from time-to-time in the application of FDS is excessive processing time. Some simulations can take days or weeks to produce useful results even with refined (simplified) models. This is problematic for real-world design projects which may require many simulations to be run in a relatively short time frame to meet project schedules.

While many fire engineering companies have access to at least modest computational resources for FDS computation these may not be able to process FDS simulations in a useful timeframe for commercial projects. Under these circumstances the use of contracted High Performance Computer (HPC) processing may provide the required computational resources to meet contractual time frames.

There are numerous publications about FDS run-time using parallel processing[2,3], a number of which make run-time comparisons between different hardware platforms. However these are not directly applicable to the hardware platforms considered in this project. Further, both the computer hardware, required for parallel processing software, and FDS continue to evolve. A new comparative study of FDS performance is therefore warranted.

This study is largely a book keeping exercise and is therefore of limited academic merit. However fire engineering practitioners may find the results from over 8,000 core/hours of processing useful for assessing project computational requirements and reducing model simulation time.

It should be noted at the outset that parallel processing of FDS models is not a panacea for improving model processing time. Model refinements and the selection of an appropriate computational domain may be significantly more productive.

While many models can be expected to run successful in a parallel processing environment with a reduction in run-time some will fail to run to completion – usually, but not always, associated with a weakness in the model. Resolving parallel processing issues can be extremely difficult and time consuming.

## 2.1 Note of Appreciation

FireNZE thanks New Zealand eScience Infrastructure (NeSI) and The Centre for eResearch, University of Auckland for providing computer processing time on the HPC Pan cluster for this project.

I would like to acknowledge the assistance provided by Mr Gene Soudlenkov (NeSI Support) for his prompt and helpful responses throughout this project, and the fire engineers and academics that took time to read the draft and provide critical comments and suggestions.

## 3 Purpose

The purpose of this project is to compare the processing speed of FDS on an HPC platform against more modest computational resources.

The evaluation will also consider the run time performance of MPI and OMP parallel processing strategies, aspects of model optimization for improvements in run-time, and other factors influencing processing performance and model results.

# 4 Discussion - FDS Processing Speed

## 4.1 Amdahl's Law

The parallel processing of FDS is generally described by Amdahl's law[4] with a serialization percentage between about 20 and 60%[5]. A direct consequence of this is that there are asymptotic limits to the maximum processing speed that can be achieved through increased computational parallelization as shown in Figure 1.

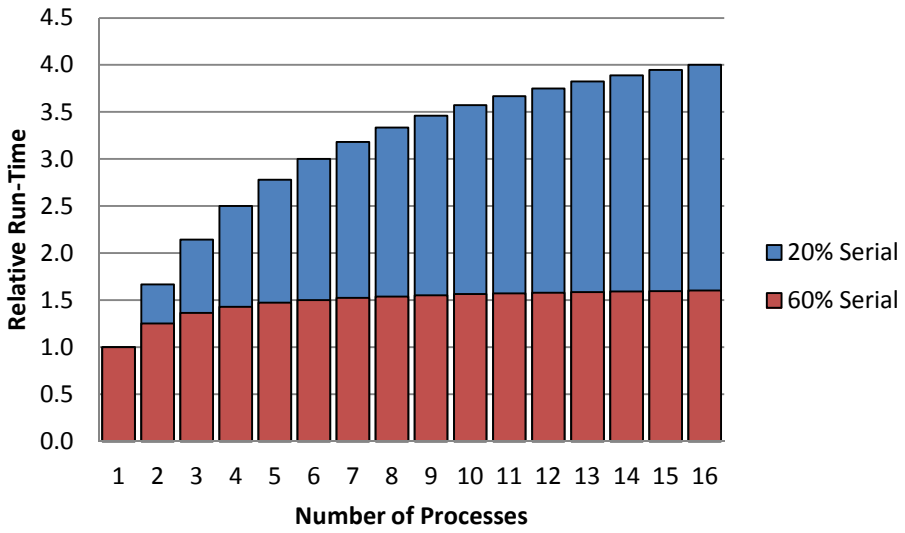


Figure 1. Amdahl's Law for FDS

The point of diminished returns for parallel processing of an FDS model typically occurs with allocation of between 8 and 16 processes. Run-time reductions through the application of more than 16 processes to a single simulation can be expected to be minimal.

## 4.2 MPI and OMP

The FDS environment provides two methods of parallel processing called OMP (Open Multi-Processing, also referred to as OpenMP) and MPI (Message Passing Interface). The two methods can be applied concurrently on the same FDS simulation, subject to hardware limitations.

OMP allows an FDS model with one or meshes to be run in parallel on a *single CPU (Central Processing Unit)* with one or more cores. While OMP can be run concurrently on several CPU's it cannot run the same process in parallel on different CPU's. Section 3.1.1 of the FDS Users Guide[6] suggests that the maximum speed benefit of OMP will be about a factor of two.

MPI allows an FDS model *with more than one mesh* to be run in parallel on more than one CPU on one or more computers. MPI will almost always be more productive than OpenMP[6, Sect. 3.1.3].

## 4.3 Hyper-Threading

A number of computer processors[7] offer hyper-threading where a single physical processing core is allocated to several virtual cores through embedded firmware within the core. The use of virtual multi-threading has been shown to be detrimental[3] to the overall processing time of FDS.

In this project virtual multi-threading was disabled so that computational processes were assigned to physical processing cores.

## 4.4 Other Run-Time Improvement Strategies

### 4.4.1 CPU Speed

For a given processor architecture CPU clock speed is a key parameter for FDS run-time. An FDS model running on a computer with a 4.4 GHz processor clock speed can be expected to complete significantly faster than on a 2.7 GHz computer. The actual speed improvement is unlikely to be a linear function of processor speed due to other factors such as memory access time.

### 4.4.2 Turbo and Over-clocking

Some CPUs have the facility to modestly increase clock speed using turbo modes or over-clocking without radical modification to cooling. While these modest speed improvements may be useful for improved processing performance they come at the expense of potential instability, voided warranties and a reduction in anticipated hardware service life. Long term processor stability should be evaluated after application of clock speed enhancements using programs such as Prime95[8].

### 4.4.3 CPU Architecture

CPU architecture is also important to FDS processing speed. The bandwidth and size of processor cache memory (fast on-chip memory used for transfer of information between cores) can be expected to have a significant influence on the speed of parallel processing.

Note that the CPU speed of servers (as typically used in HPC clusters) will generally be lower than for desktop computers, but with greater amounts of higher bandwidth cache. This is a consequence of the typical usage, power efficiency and reliability required in a server environment.

### 4.4.4 Network

Network speed is an important consideration in parallel processing applications where processors on different nodes need shared access to dynamic data.

Under MPI it is possible to tune network performance for improved data transfer between FDS processes on separate computers. There are also a number of different protocols for data transfer across a network that offer different levels of network speed and security (cf. SSH and RSH[9], rsync and mpisync). Network tuning and data transfer protocols will not be specifically evaluated through this project.

### 4.4.5 Processing Overheads

Operating System (OS) and other application over-heads are relevant to FDS computational performance. This project does not examine these aspects of the computational environment on FDS run-time performance. Other software under user control that is not necessary for FDS processing or project data collection was not run during FDS simulations.

Similarly, the run-time priority (niceness in Linux parlance) assigned to FDS processes by the OS was retained at defaults. Modest processing speed improvements can be realised by increasing the run time priority of FDS and associated software through the OS, but with a risk of introducing stability issues.

## 4.5 Computational Domain Size

Most FDS users will be aware that the computational time required for a simulation increases significantly with increasing resolution of the computational domain.

The computational burden for an FDS model will increase by a factor of about 16 when the computational domain basic cell dimension is halved (increasing the number of computational cells by a factor of 8). This is a direct consequence of scaling in a three dimensional orthogonal Cartesian coordinate system (but note that FDS models can be configured in two dimensional and axially symmetrical cylindrical space)[6, Sect. 6.3.2].

The size of the computational domain must clearly encompass the fire cells of interest and will often extend to the entire building and beyond. Subject to the symmetry of the problem it may be possible to use mirror boundary conditions to minimize the computational domain.

Note: In practical modelling it may be necessary to extend the computational domain beyond the building envelope to ensure vent flows are appropriately simulated. Extending the model also allows rapid visual assessment of inadvertent leakage paths through the specified geometry.

## 4.6 Grid Resolution

Grid resolution is an important FDS parameter in determining how well a model will represent the fire phenomena of interest, and directly affects the size of the computational domain.

Grid resolution is initially determined by rules of thumb:

> *The design fire's equivalent diameter should be spanned by between 10 and 18 basic cubic cell dimensions.*

or through consideration of the parameter D*/dx, but should be confirmed through sensitivity analysis by examining metrics of interest with increasing resolution.

FDS provides turbulence and scalar resolution metrics that can provide insight into the appropriateness of the grid resolution. These metrics are seldom used in practise. Recent releases of FDS (later than 6.2.0) have resulted in changes to the availability turbulence and scalar resolution[10].

## 4.7 FDS Model Parameters

There are many user-adjustable FDS parameters that can lead to dramatic changes in processing time for a given simulation. These include the introduction of particles, radiation angular resolution, the specification of pressure zones and HVAC. Those aspects of an FDS calculation that are not important to the validity of a simulation, or will not contribute useful output, should be dispensed with.

For example the calculation of radiation from a relatively small fire in a very large space may have little influence on fire growth through compartment effects and have no measurable influence on tenability. In this instance if FDS processing speed is an issue then the radiation solver could be disabled[6, Sect. 1.4 (3)].

This study cannot reasonably assess the effects of every FDS parameter on model run-time. Nor are we particularly interested in the appropriateness of model resolution, other than to ensure that model output is comparable between defined simulations on different computational platforms.

# 5 Project Methodology

The project examines the run time of a relatively simple FDS model with different mesh configurations and resolution on three hardware platforms.

- Each test case was run to completion using identical process allocations.
- Run-time was measured and compared between platforms, allocated processes and the method and extent of applied parallel processing.
- Model output was compared between platforms, allocated processes and the method and extent of applied parallel processing.
- Run-time and model output variance was examined.

# 6 Computational Platforms

The following hardware platforms were selected for this project.

- HPC Pan Cluster
- Beowulf Linux Cluster
- Windows Workstation

## 6.1 HPC Pan Cluster

The HPC Pan cluster[11] is a research and commercial resource provided by The Centre for eResearch, University of Auckland through New Zealand eScience Infrastructure (NeSI)[12].

The Pan cluster is capable of concurrently running hundreds of FDS models with significant parallelization (> 6,000 physical cores, but subject to allocated resources and concurrent use by other subscribers).

The Pan cluster includes a variety of Intel architecture processors running at between 1.87 and 2.8 GHz. It has significant RAM resources and is networked at 40 Gb/s. Test cases run on the Pan cluster were run in Node Groups b, c, d and e with Intel Xeon E5-2680 cores running at 2.7 GHz. The processors run Red Hat Enterprise Linux 6.3 OS.

The cluster is organised into node groups of identical architecture nodes. The resources of nodes vary from group to group which may result in changes to the resources allocated to a particular simulation, subject to node group allocations in the submission batch file.

## 6.2 Beowulf Cluster

A Beowulf[13] cluster was built by FireNZE as a low-cost dedicated solution for fast processing of a limited number of FDS models with limited parallelization (up to 16 cores per model).

The cluster comprises four Intel I7 4097K quad core processors, each with a clock speed of 4.4 GHz, 16 GB RAM and solid state hard drives, connected with a dedicated 1 Gb Ethernet LAN. The machines run under Linux Ubuntu 14.04 64 bit OS.

## 6.3 Windows Workstation

This is a typical personal computing platform that exceeds the minimum recommended hardware specifications for FDS processing[6, Sect. 2.2 et seq.]. The platform has a single CPU and cannot run FDS in an MPI parallel environment.

Effective concurrent modelling capability is limited to the allocation of four cores under OMP.

The platform comprises an Intel I7 3770 quad core processor running at 3.9 GHz with 16 GB RAM.  The machine runs under a Windows 7 64 bit OS.

# 7 Project FDS Model

The test model used for this project, while representative of a typical FDS analysis of a compartment fire, is entirely fictional. The FDS input file is listed in Appendix A.

## 7.1 FDS Version

The model was processed on all hardware platforms using FDS Version 6.2.0, Compilation Date: Sat, 11 Apr 2015, SVN Revision: 22343. This was the most recent release of FDS at the time this project was initiated.

### 7.1.1 Discussion

As a general rule practitioners should use the most recent release of FDS when commencing a new project and, in my opinion, regulators are right to insist upon this.

The reasons for upgrading include improvements in functionality and coding, enhancements to the calculation of the underlying physics (noting that the physics do not change) and the provision of software support. Version changes are fully documented in the FDS release notes that accompany the software and all releases are subjected to validation[18] by NIST.

There was a general fire industry reluctance to upgrade from FDS Version 5.6 to Version 6.0 because of increased processing burden (the very subject of this study). It is apparent from NIST's change management process and the FDS User's Manual[6] that the development team were cognisant of this issue. Clearly they considered that the upgrade enhancements more than compensated for the increased run-time.

Adoption of the latest release of FDS also provides the impetus for continued development and support of the program.

## 7.2 Geometry

All simulations are based on an 8 m wide x 16 m long x 4 m high compartment model with two open vertical vents, each 2 m high x 0.75 m wide as shown in Figure 2.

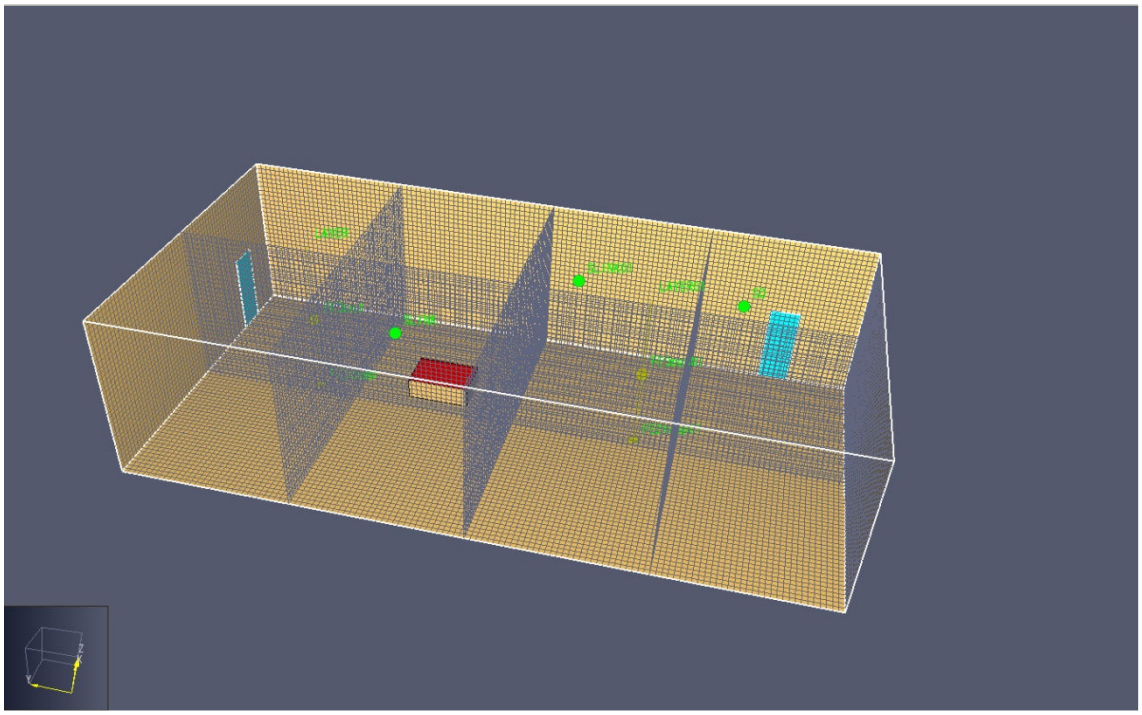

Figure 2. Test Model (8 mesh, 0.125 m cubic cell dimension)

## 7.3 Computational Domain

The model computational domain was established using a single-sized cubic cell (mesh stretching was not applied). The cell basic dimension was varied as shown in Table 1. For brevity these are referred to as the 32K, 260K and 2M computational domains throughout this report.

The range of computational domain sizes tested in this project are considered to be small to moderate. Large computational domains might extend to tens of millions of cells.

| Cell Size (m) | Computational Domain (Number of Cells in Model) | Abbreviation |
|---|---|---|
| 0.25 | 32,768 | 32K |
| 0.125 | 262,144 | 260K |
| 0.0625 | 2,097,152 | 2M |

Table 1. Model Computational Domains

Initial experiments with a cell size of 0.125 m and a single mesh required about three hours of processing for a 180 second simulation on the Windows Workstation. This was considered to be an adequate duration to ensure that FDS initialization and wrap-up processes were not a significant contribution to the total computation time. The single process 2M case was expected to run to completion in approximately 72 hours.

The design fire was offset from the centre of the compartment to ensure that remained completely bound within a single mesh for the multiple mesh test cases. Compartment dimensions were selected to ensure that the design fire plume was largely contained within a single mesh in the multiple mesh test cases. The rationale for this is to reduce high levels of model activity at inter-mesh boundaries. This is considered to be good practise for real world modelling problems[6, Sect. 6.3.4].

The compartment ventilation area was sized to ensure that the design fire remained fuel-controlled throughout the 180 second simulation, but with significant hot upper layer development to ensure well-distributed fire phenomena for defined meshes.

## 7.4 Meshes

Multiple meshes are required for the application of MPI parallel processing.

Test models were evaluated with 1, 2, 4, 8 and 16 meshes. Each mesh in any particular test case contained an identical number of computational cells. The number of assigned MPI processes for a test case was equal to the number of meshes.

Mesh cell dimensions were established as factors of 2, 3 and 5 to ensure optimized processing[6, Sect. 6.3.1]. Note that this arrangement does not necessarily provide for optimum distribution of the computational burden between meshes, particularly early in a simulation.

Mesh priority for abutting meshes of the same resolution is understood to have no affect run-time under recent FDS releases[14]. The general advice on this issue is that finer resolution meshes should have a higher priority (be listed earlier in the FDS input file) than courser meshes, otherwise mesh priority is not important. Mesh priority in this project was assigned on the basis of the anticipated mesh of most fire activity to the least activity (old habits die-hard).

## 7.5 Design Fire Specification

The design fire was specified with a fast $\alpha t^2$ growth rate, a plan area of 1.5 $m^2$, and a maximum Heat Release Rate Per Unit Area (HRRPUA) of 1,000 $kW/m^2$. The model was run to simulate 180 seconds with a final heat release rate of 1.55 MW.

The design fire was specified in accordance with C/VM2[15] requirements for occupancies with less than 3 m of storage height. Combustion yields were established in accordance with C/VM2 using defined chemistry in the FDS single stage combustion model.

## 7.6 FDS Parameters

The initial FDS flow field[6, Sect. 6.4.1] was defined without noise (&MISC NOISE=.FALSE.) on the misunderstanding that the pseudo-random flow-field noise seed was not constant and would result in variations in results and run time for a particular model. Subsequent analysis showed that the seed is constant resulting in consistent model output and no contribution to variations in model processing time. In real world projects this parameter is .TRUE. by default to prevent the development of a perfectly symmetrical flow field in a symmetrical domain – fire is, after all, a stochastic process.

With the exception of NOISE=.FALSE., FDS simulation parameters were run at default values.

## 7.7 Model Output

In order to exercise computer platform data storage a number of typical FDS output parameters have been defined in the model as follows:

- Temperature
- Visibility
- Velocity
- FEDco
- Radiation
- Upper Layer Parameters
- Turbulence Resolution
- Scalar Resolution

These measurements are to be recorded as plot files, slice files, boundary files, point and linear measurements (as appropriate).

Restart files (&DUMP RESTART = 30) were also recorded every 30 simulation seconds. Restart information is not essential for this particular project as the model runs to completion is a reasonable timeframe and individual results are not project critical. Although the restart files are relatively large and exercise disk access they are not an appreciable contributor to total run-time.

Restarts are recommended for medium to large FDS simulations for commercial purposes to minimize the extent of reprocessing in the event of computer outage. They can also provide useful information for processing computational errors (such as numerical instability) and allow some aspects of the simulation to be adjusted through the course of a calculation.

## 8 Project Test Cases

The project test cases are described in Tables 2 and 3. Each test case was run on each computer platform with the proviso that the physical processing cores were not over-subscribed (not more than one computational process was allocated to a single physical core).

| Model Number | Meshes | MPI Processes | OMP Processes | Cores | Cell Size |
|---|---|---|---|---|---|
| F20-1-1 | 1 | 1 | 1 | 1 | 2.0 |
| F10-1-1 | 1 | 1 | 1 | 1 | 0.125 |
| F05-1-1 | 1 | 1 | 1 | 1 | 0.5 |

Table 2. Mesh Resolution Run-Time Evaluation

The mesh resolution tests in Table 2 provide a comparative measure of single process computational speed. These provide the single-process bench-mark for the parallel processing cases.

The applied cell sizes seeks to confirm the 16 times increase in processing time with a halving of the basic computational cell dimension, and provides a metric for estimating the run-time of other models.

| Model Number | Meshes | MPI Processes | OMP Processes | Cores |
|---|---|---|---|---|
| Mxx-1-2 | 1 | 1 | 2 | 2 |
| Mxx-1-4 | 1 | 1 | 4 | 4 |
| Mxx-1-8 | 1 | 1 | 8 | 4* |
| Mxx-2-1 | 2 | 2 | 1 | 2 |
| Mxx-2-2 | 2 | 2 | 2 | 4 |
| Mxx-2-4 | 2 | 2 | 4 | 8 |
| Mxx-2-8 | 2 | 2 | 8 | 8* |
| Mxx-4-1 | 4 | 4 | 1 | 4 |
| Mxx-4-2 | 4 | 4 | 2 | 8 |
| Mxx-4-4 | 4 | 4 | 4 | 16 |
| Mxx-4-8 | 4 | 4 | 8 | 16* |
| Mxx-8-1 | 8 | 8 | 1 | 8 |
| Mxx-8-2 | 8 | 8 | 2 | 16 |
| Mxx-8-4 | 8 | 8 | 4 | 16* |
| Mxx-8-8 | 8 | 8 | 8 | 16* |
| Mxx-6-1 | 16 | 16 | 1 | 16 |
| Mxx-16-2 | 16 | 16 | 2 | 32* |
| Mxx-16-4 | 16 | 16 | 4 | 64* |
| Mxx-16-8 | 16 | 16 | 8 | 128* |

Notes: * Cores may be over-subscribed subject to hardware.
xx replaced with mesh resolution and hardware platform identifier

Table 3. MPI and OMP Run-Time Evaluation

The MPI and OMP tests are designed to comparatively evaluate parallel processing configurations within and between hardware platforms.

Two further sets of 100 simulations were run to measure the simulation time variability for a given model on the Beowulf Linux cluster. This also provided insight into OS overheads and processing bottlenecks on this hardware platform.

The model set for the project comprised 46 test model simulations, a further 200 simulations completed to evaluate run time variation, and a number of developmental simulations. The total processing time for the project across all platforms exceeded 8,000 core hours.

# 9 Project Implementation

The base FDS test model was prepared (refer to Appendix A) and tested.

The base model was reconfigured for multiple meshes and resolutions and run under different process allocations on each of the three hardware platforms over a period of 16 weeks.

A template script file was provided by NeSI for running the test cases on the HPC platform under SLURM[16] (Simple Linux Utility for Resource Management).

A further set of analyses was completed on the Linux cluster to examine test case run-time variability. Two models were each run several hundred times under a Linux shell script that extracted model run-time and DEVICE activation time as a measure of variability.

## 9.1 Post Simulation Processing

Post-simulation processing of output data, as might be required for a fire engineering design, is outside of the scope for this project other than for comparing model results between hardware platforms.

The size of the FDS output files for the refined mesh models was substantial, extending to approximately 4 GB for the 2M models. Downloading the FDS output files from the HPC Pan cluster for local analysis proved to be a significant burden at ADSL speeds (measured at typically 0.23 MB/s). Even with VDSL and an assumed increase in download speed by a factor of ten to 2.3 MB/s, the time for FDS model download can be an appreciable fraction of processing time - perhaps as much as 6%.

In order to reduce the download burden the entire FDS output of only four test cases were downloaded from the HPC Pan cluster in their entirety. All aspects of these complete datasets were compared with other platform test cases to validate the model on each platform.

For all other test cases downloads were limited to the following subset of FDS and SLURM output files.

FDS Model
Device File
HRR File
Model Output File

SLURM Runtime Script
SLURM Error File
SLURM Output File

This minimal download was sufficient to examine test case run-time, compare model heat release rate history and device activation times, confirm allocated computational resources and ensure that each test case had run to completion without error.

Test case run-time and device activation measurements were subject to comparative analysis across platforms and between applied computational resources; the results summarised, considered and reported.

# 10 Results and Analysis

The FDS results of the simulated models extends to 100's of GB on three hardware platforms. This project is primarily concerned with simulation run time so the bulk of this information (which would be essential to understanding the fire performance of the model) is redundant and is not included in this report.

Run-time data has been reduced to comparative graphics for ease of visual interpretation. The reader is cautioned to note Y axis scales when comparing graphics.

The data is necessarily sparse because simulation of all possible parallel processing recourse allocations would be a formidable, expensive and time-consuming task, even with limited computational domains.

The bulk of the experimental data is presented as discrete MPI and OMP resource allocations plotted against run-time. Trend lines have not been incorporated in the graphs because the data is generally sparse and because these do not accurately portray discrete events.

## 10.1 Single Mesh with OMP

Figures 2 to 4 show test model run-time on the three (Windows, Pan and Beowulf) hardware platforms with increasing OMP parallelization and increasing mesh resolution (the progressively larger computational domains in Table 1).

The Pan platform provided an anomalous (extended) run-time for the OMP = 2 case on the 32K computational domain (refer to Figure 2). The cause could not be identified and the model could not be readily re-run due to limitations of project processing allocation.

### 10.1.1 Processor Speed

It is immediately evident from Figures 2 to 4 that increasing processor speed translates directly to decreased model run-time.

For the OMP Processors = 1 case there is no parallel processing and the variation of run time between platforms decreases with increasing processor speed. This is independent of the size of the computational domain. The Beowulf platform (4.4 GHz) was 21% faster than the Windows platform (3.9 GHz), and 47% faster than the Pan platform (2.7 GHz).

The relationship between processor speed and run time is not linear. This is considered to be a consequence of differences in the hardware and software configurations of the three platforms.

### 10.1.2 OMP Parallelization

The extent of this analysis on the Beowulf and Windows platform was limited to 4 OMP processes by the available processor cores.

Application of increasing OMP parallelization (up to 8 OMP processes) resulted in reduced model run-time, but with decreasing returns. This is in general accordance with Amdahl 's Law.

The reduction in run-time with increasing OMP parallelization improved with increasing computational domain size. This is attributed to a relative increase in model calculation time verses parallelization burden.

The application of more than 8 OMP processors on the Pan platform resulted in an increase in model run-time. This is attributed to the serialization percentage of the model and the increasing parallelization burden.

The Pan cluster provided the greatest percentage reduction in run-time through OMP parallelization with 8 OMP processes on the 2M computational domain (Figure 4). The run-time improvement was 54% reducing from 89.9 hours to 39.9 hours. The maximum run-time reduction with OMP parallelization was typically less than 50% across all model resolutions and hardware platforms.

The reason why the Pan platform is more efficient in the application of OMP than the Windows or Beowulf platforms is attributed to its core cache size and bandwidth. The Pan Xeon processors have a 20 MB cache with a memory bandwidth of 51.2 GB/s compared with Intel I7 processors with 8 MB cache and 25.6 GB/s bandwidth.

With the application of OMP the Beowulf platform provided the best run-time, being approximately 30% faster than the Windows and Pan platforms.

### 10.1.3 Run-Time and Computational Domain Size

As the computational domain size increases by a factor of eight (the basic cell dimension is halved progressing from Figure 2 to 3 and from Figure 3 to 4) the model run-time increases by a factor of approximately 16. This result applies to all platforms and is independent of OMP or MPI parallelization.

Model run-time can therefore be predicted by initially running a low resolution (and relatively fast) simulation and scaling accordingly. Initial low resolution modelling also provides a useful check on aspects of FDS model functionality, including device triggered events, design fire growth rate, model leakage and stability and is therefore recommended practise.

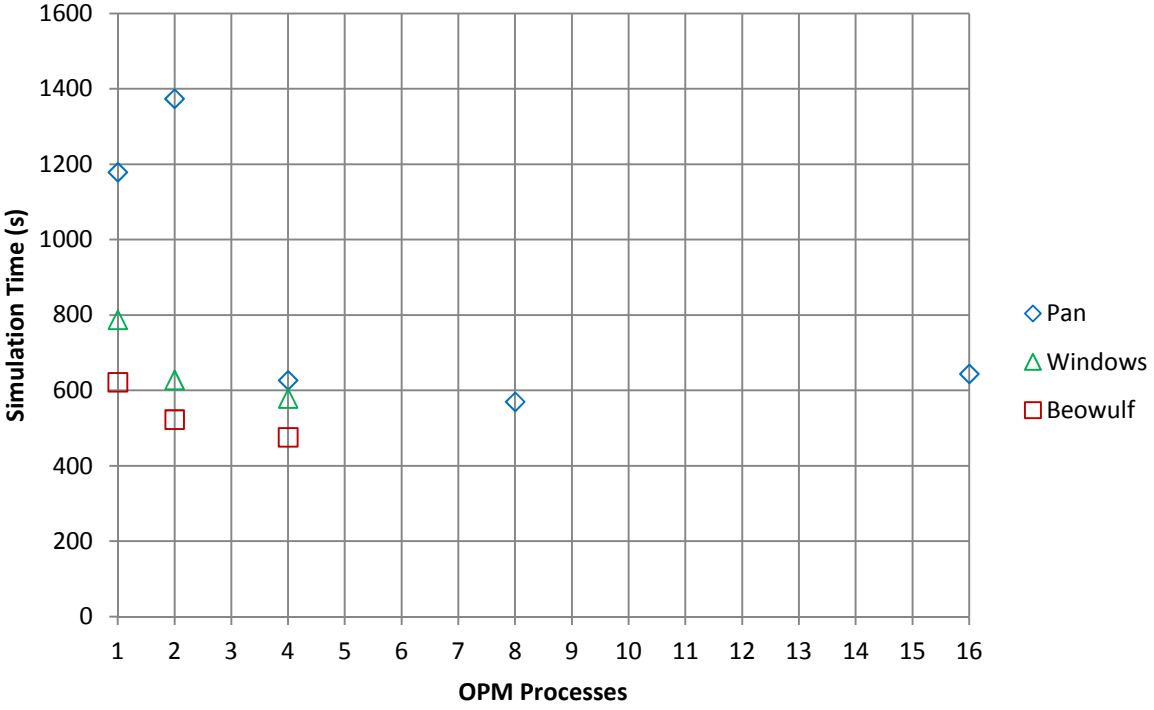


Figure 2. Simulation Time with Increasing OMP Parallelization
1 MPI Process (1 Mesh), 32K Computational Domain

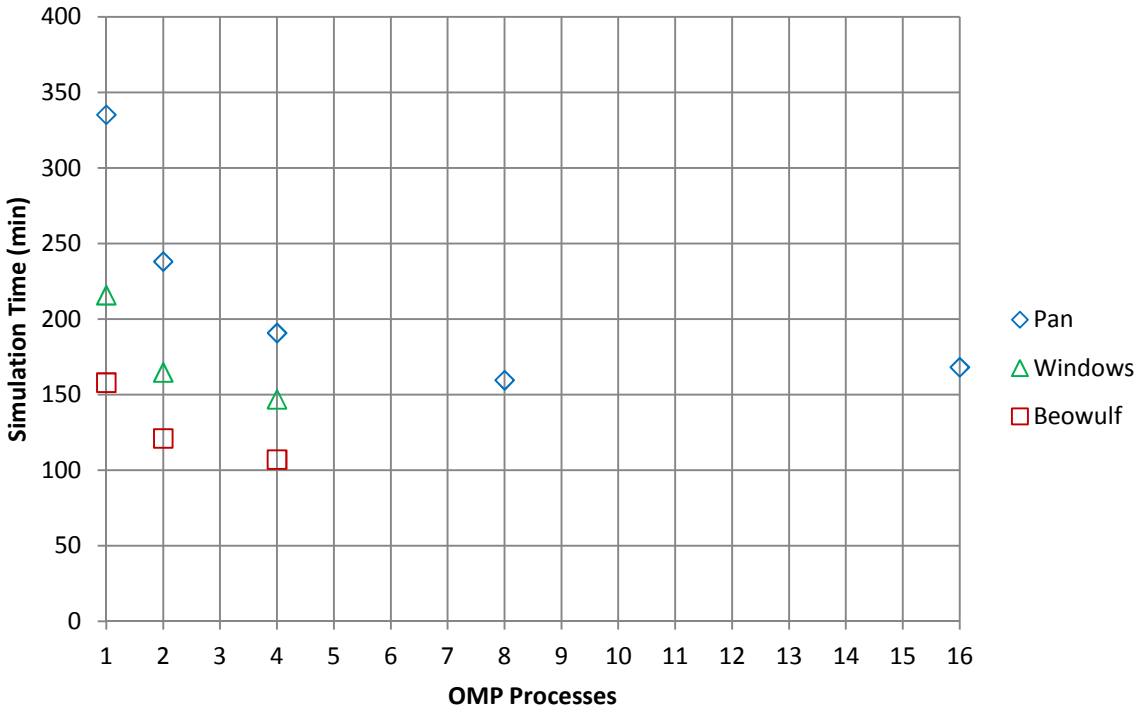


Figure 3. Simulation Time with Increasing OMP Parallelization
1 MPI Process (1 Mesh), 260K Computational Domain

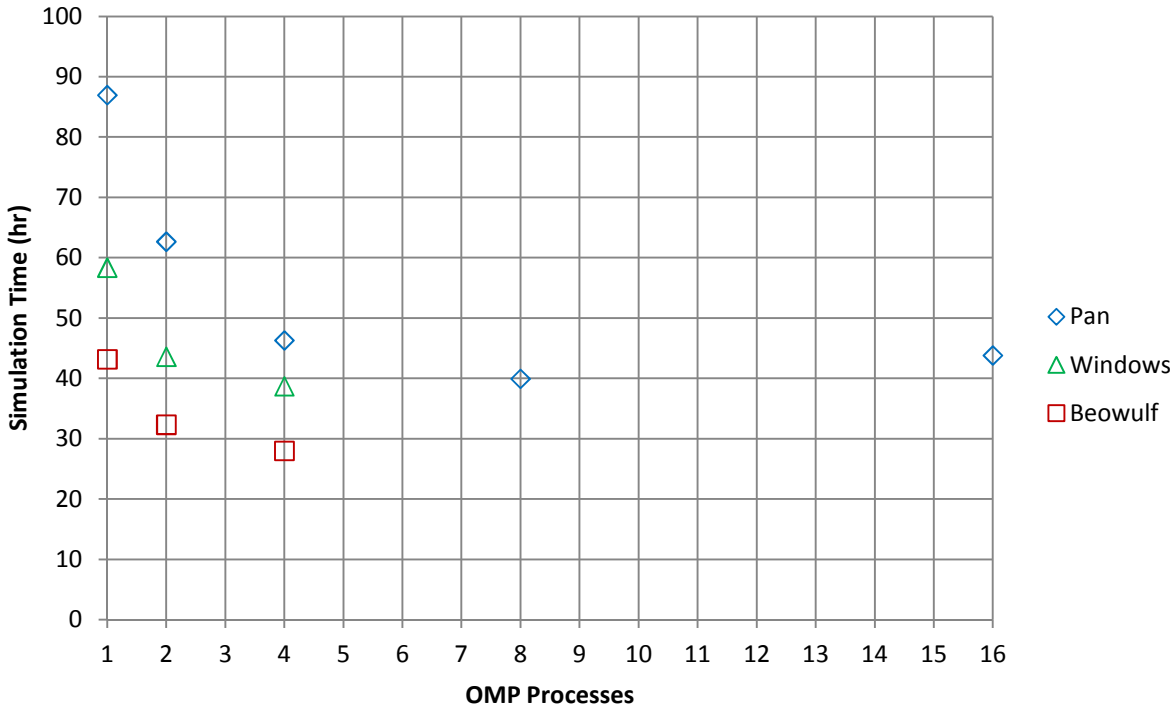


Figure 4. Simulation Time with Increasing OMP Parallelization
1 MPI Process (1 Mesh), 2M Computational Domain

## 10.2 MPI (Multi-Mesh)

Figures 5 to 7 show the model run-time difference between the Pan and Beowulf hardware platforms with increasing MPI parallelization and increasing mesh resolution (the progressively larger computational domains in Table 1).

Note that MPI cannot be applied on the Windows platform because it has a single 4-core processor. Although multiple mesh models can be run on the Windows platform these cannot be invoked with MPI parallelization.

### 10.2.1 Processor Speed

Processor speed remains a significant factor for reduced run-time, particularly with low numbers of applied MPI resources and larger computational domains.

### 10.2.2 MPI Parallelization

Run-time decreases (improves) with increasing MPI parallelization on both hardware platforms, but with decreasing returns. This is an expected result in general accordance with Amdahl 's law.

For larger computational domains the Beowulf platform was fractionally faster than the Pan platform with 16 MPI processes.

Both the Beowulf and Pan platforms were significantly faster than the Windows platform under OMP with at least an 80% reduction (improvement) in processing time (38.7 hours reduced to 6.6 hours for the 2M model).

For a given number of cores MPI parallelization resulted in faster run-times than OMP (compare Figures 2 and 5, Figures 3 and 6 and Figures 4 and 7). For example the run-time improvement of the 2M model on the Pan platform from 2 to 4 OMP processes (Figure 4) is 26%, while the run-time improvement from 2 to 4 MPI processes (Figure 7) is 46%.

The rate at which MPI parallelization reduces run-time is significantly greater on the Pan platform, to the extent that the speed advantage of the Beowulf platform is negated after the application of 4 MPI processes on the 32K computational domain model and 16 MPI processes on the 260K and 2M cell models.

This is attributed to the cache bandwidth described previously and the network speed of the Pan platform (40 Gb/s) compared to the relatively slow 1 Gb/s of the Beowulf platform.

Examination of the Beowulf Ethernet network load during MPI model processing showed a average data transfer rates of approximately 30 MB/s (concurrent send and receive) between computer nodes. The time required for data transfer on the Beowulf network is therefore a significant proportion of the total time that might otherwise be available for model calculation. Approximately 25% of model processing time is required for data transfer between nodes (30 $MB/s_{duplex}$ x 8 b/B / 1 Gb/s = 0.24). Assuming that the Pan platform requires similar data transfer rates between nodes for MPI processing then the network overhead is less than 1%.

This identifies that the 1 Gb/s Ethernet throughput is a significant deficiency with the Beowulf hardware for MPI processing. While hardware improvements to the network such as Infiniband or optical LAN networking would be expected to improve the Beowulf MPI processing time these are relatively expensive additions to what is a budget FDS hardware platform.

An alternative approach for MPI processing speed improvement on the Beowulf platform is improved network data transfer efficiency and protocols (see Section 4.4.2 above). Literature[17] suggests that RSH may provide at least a 20% improvement over SSH on this secure platform.

The Beowulf platform is currently fully utilized processing commercial FDS models so experimenting with network protocols could not be completed in conjunction with this project (attempting to fix what isn't broken can result in really broken with extensive, unproductive down-time).

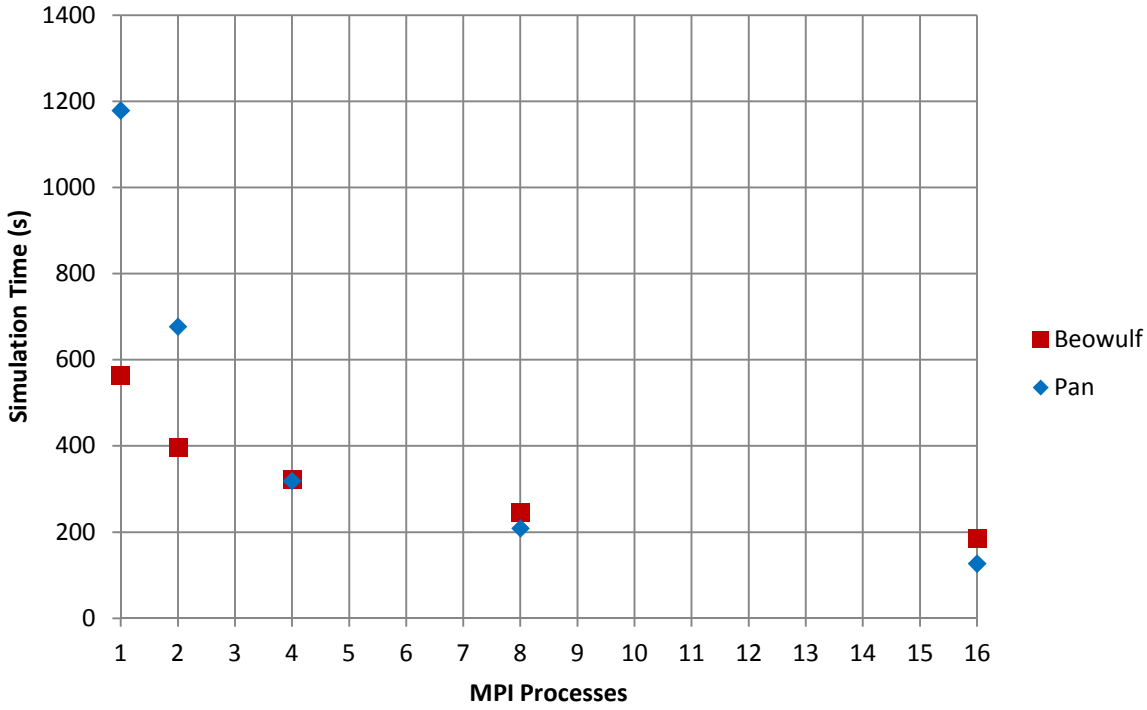


Figure 5. Simulation Time with Increasing MPI Parallelization
1 OMP Process, 32K Computational Domain

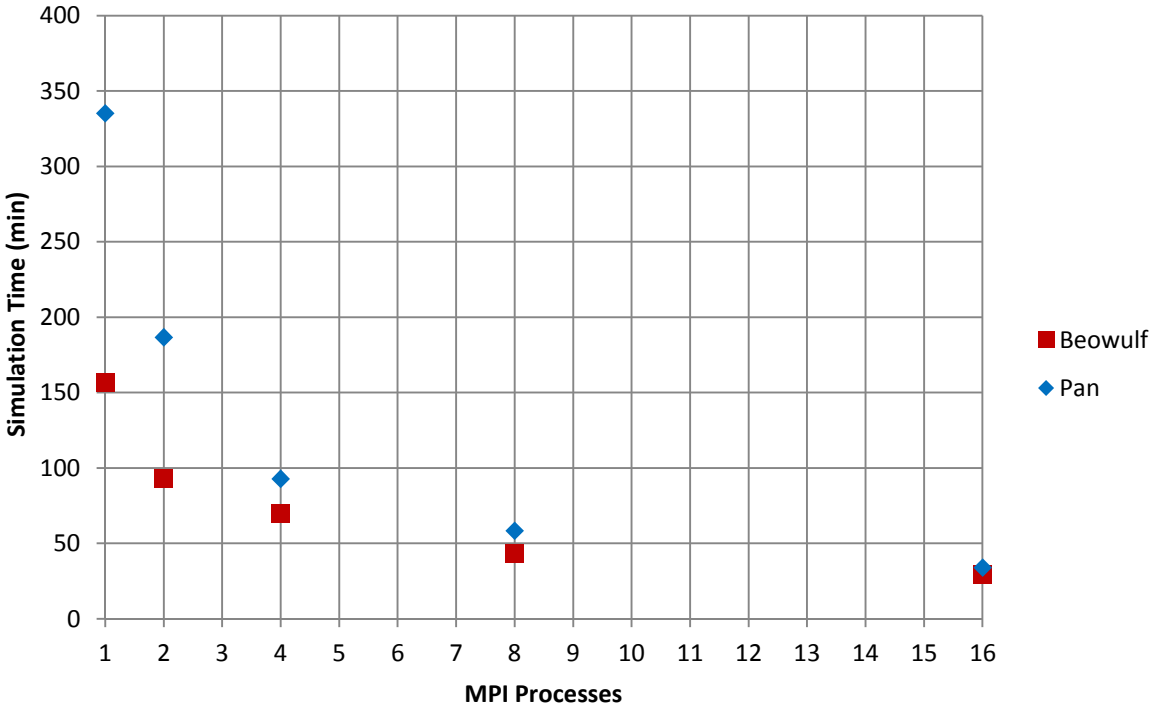


Figure 6. Simulation Time with Increasing OMP Parallelization
1 OMP Process, 260K Computational Domain

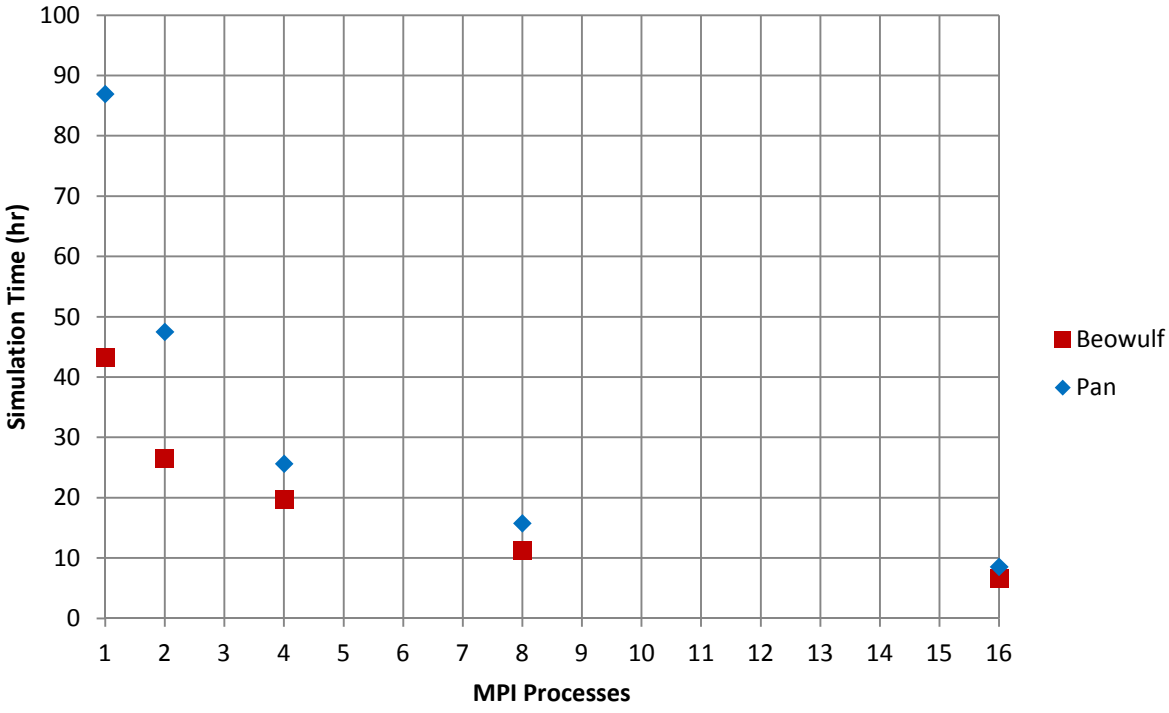


Figure 7. Simulation Time with Increasing MPI Parallelization
1 OMP Process, 2M Computational Domain

## 10.3 MPI and OMP

In preparing this report some time was spent agonizing over the most appropriate visual representation of the application of OMP and MPI. While three-dimensional graphics allow clear visualization of optimum platform performance, they do not readily show comparative performance between platforms. Also coloured information does not transfer readily to monochrome which is often necessary for publication.

For brevity this Section contains three-dimensional representations of platform performance with perspective and colour. Detailed comparative information has been included in Appendix D. Note that the graphics in this Section includes interpolated data between measured computational platform limits.

Figures 8 to 13 show the test model run-time on either the Pan or Beowulf platforms with both OMP and MPI parallelization and increasing mesh resolution. Bright yellow cells are used to identify the minimum run-time condition on a graph. The platform results for each mesh resolution are produced on a single page with appropriate scaling to facilitate visual cross-platform comparisons.

A question that one might ask is, If MPI consistently produces improved run-times than OMP, then why apply OMP? The answer is that some models are not appropriately configured for MPI (insufficient meshes or unbalanced computational loads between meshes), it may not be possible to apply available hardware resources as MPI processes, and the application of both MPI and limited OMP resources can be expected to reduced model run-times.

### 10.3.1 Platform Performance

The Pan platform out-performed the Beowulf platform for the 32K and 260K models with run-time reductions of 38% (71 seconds) and 8% (2.4 minutes) respectively, but was slower for the 2M model by 11% (0.8 hours).

The Pan platform required twice the computational resource of the Beowulf platform to achieve these results, while the differences in processing speed are arguably insignificant in the context of a real world fire engineering problems.

The minimum run-time of the test model on the Pan cluster (for each computational domain) was with a concurrent allocation of 4 OMP and 8 MPI processes. The minimum run-time of the test model was achieved on the Beowulf cluster with 16 MPI processes.

## 10.4 Run-Time Improvement

With optimum application of both MPI and OMP the best run-time improvements achieved (by platform) were:

| | |
|---|---|
| Windows: | 33.6% with 4 OMP processes |
| Beowulf: | 84.5% with 16 MPI processes |
| Pan: | 91.9% with 8 MPI and 4 OMP processes |

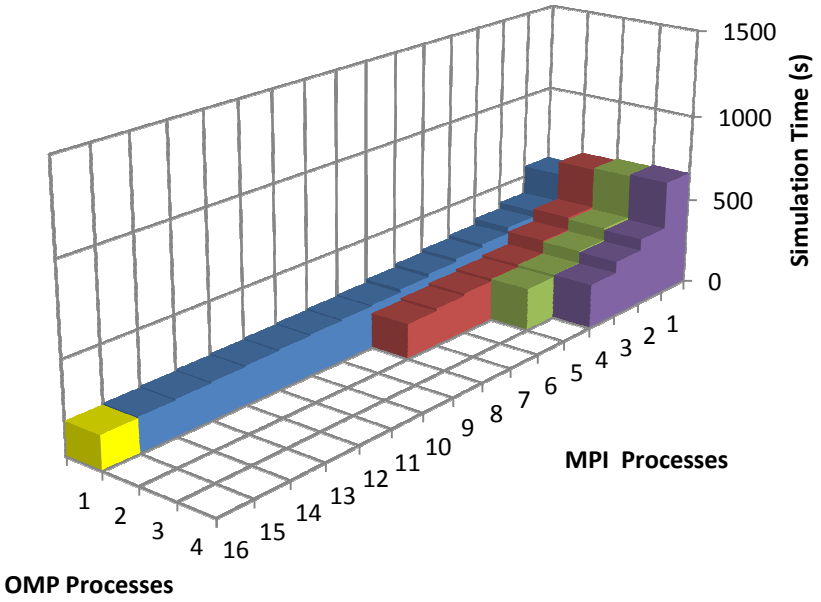


Figure 8. Beowulf Platform Simulation Time with OMP and MPI Parallelization, 32K Computational Domain.
(Minimum Run-time: 185 s with 16 MPI and 1 OMP Processes)

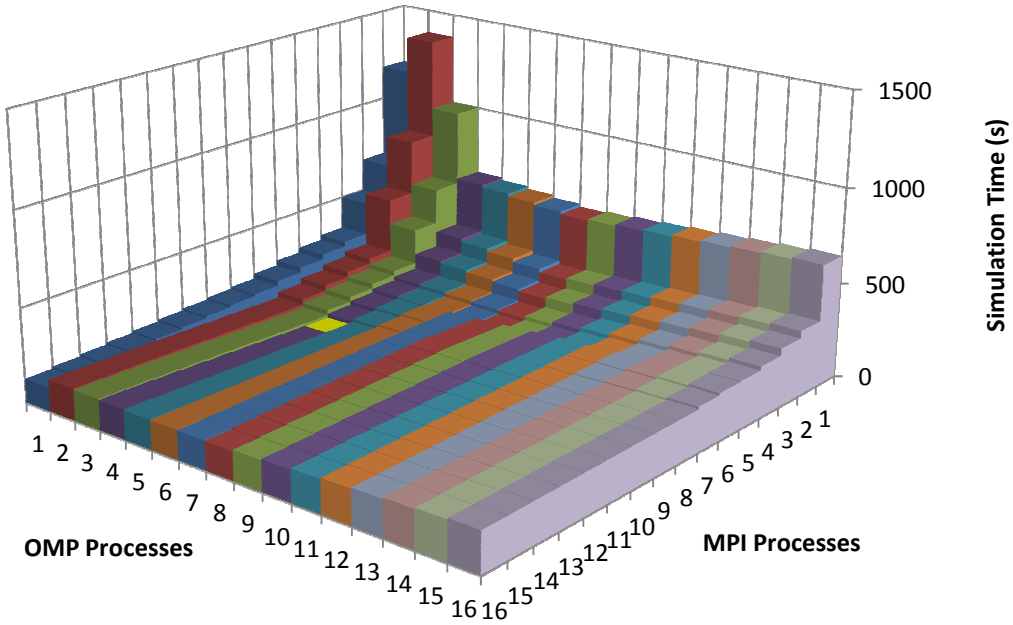


Figure 9. Pan Platform Simulation Time with OMP and MPI Parallelization, 32K Computational Domain
(Minimum Run-time: 114 s with 8 MPI and 4 OMP Processes)

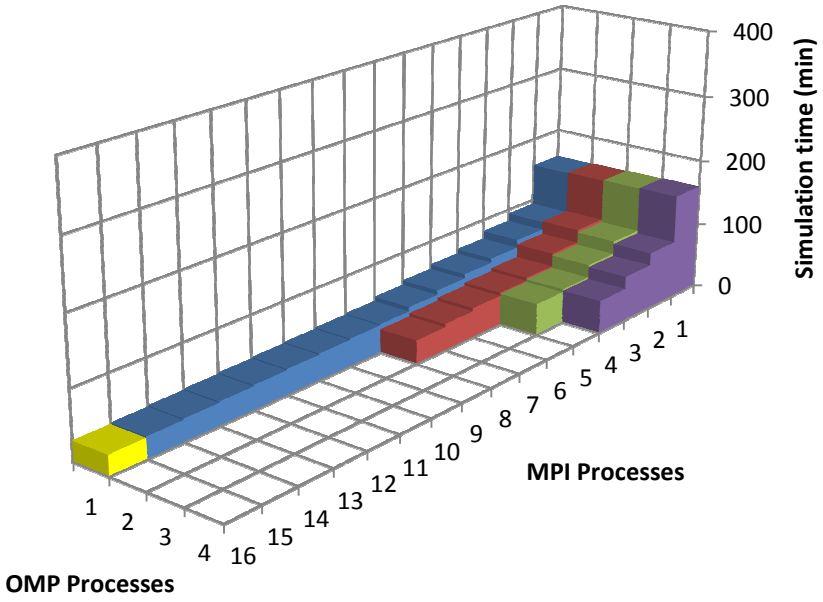


Figure 10. Beowulf Platform Simulation Time with OMP and MPI Parallelization, 260K Computational Domain.
(Minimum Run-time: 29.7 minutes with 16 MPI and 1 OMP Processes)

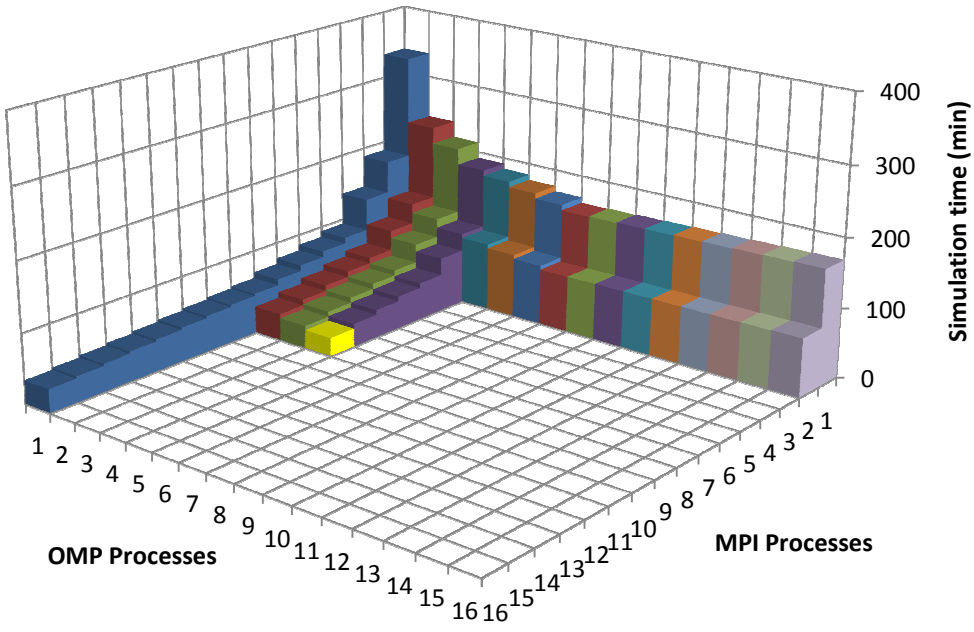


Figure 11. Pan Platform Simulation Time with OMP and MPI Parallelization, 260K Computational Domain.
(Minimum Run-time: 27.3 minutes with 8 MPI and 4 OMP Processes)

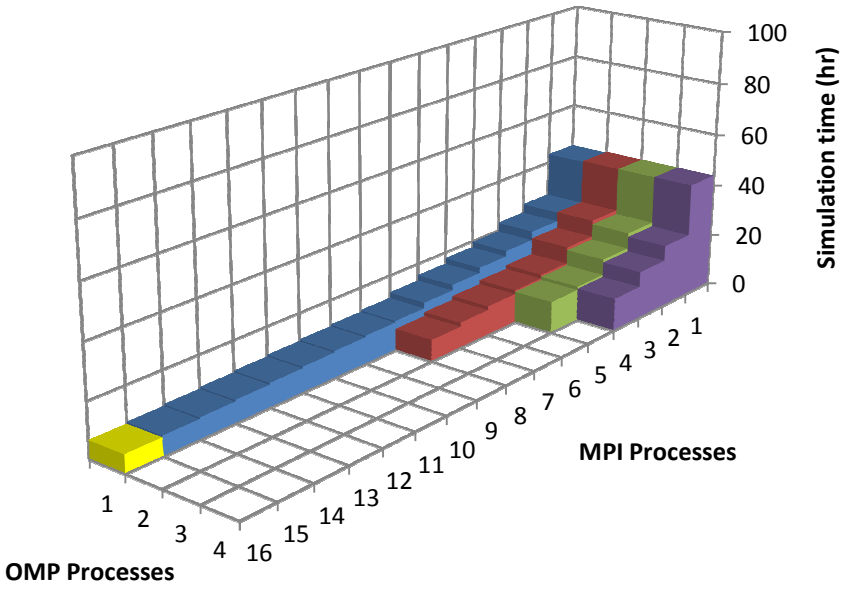


Figure 12. Beowulf Platform Simulation Time with OMP and MPI Parallelization, 2M Computational Domain.
(Minimum Run-time: 6.7 hours with 16 MPI and 1 OMP Processes)

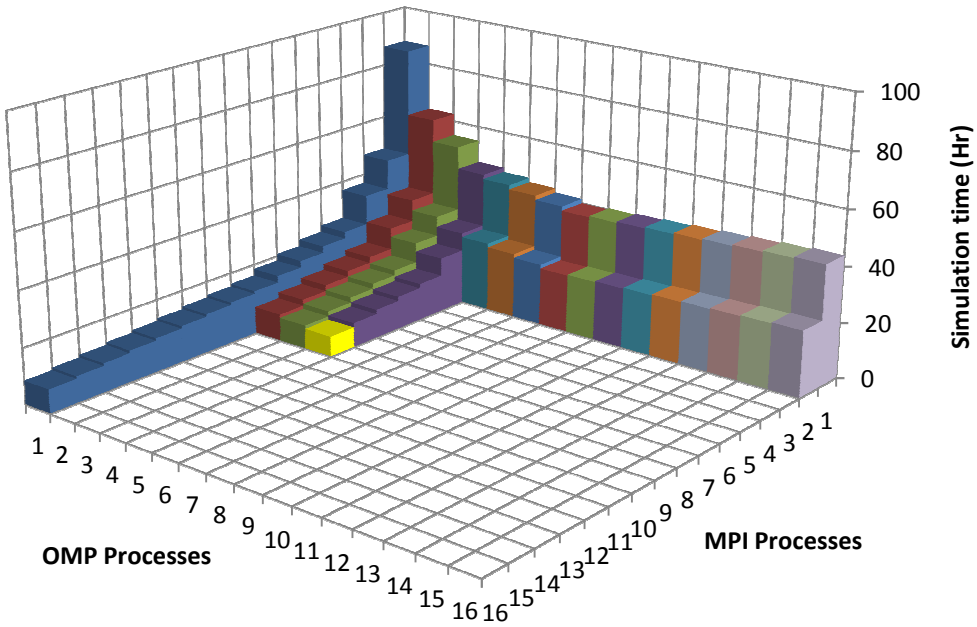


Figure 13. Pan Platform Simulation Time with OMP and MPI Parallelization, 2M Computational Domain.
(Minimum Run-time: 7.5 hours with 8 MPI and 4 OMP Processes)

## 10.5 Economy

### 10.5.1 Cost

The cost of FDS model processing is usually an insignificant fraction of the cost of a fire engineering project, and is secondary to reducing run-time to achieve project requirements.

Computer time is usually billed on the basis of expended core/hours with a commercial rate of less than US$0.20 per core hour available at the time of writing.

With efficient allocation of MPI and OMP parallelization the 2M test model was processed on the Beowulf and Pan platforms in less than 250 core/hours.

### 10.5.2 Efficiency

With the extensive processing resources available on the Pan cluster a large number of FDS models can be run concurrently, each with optimum parallelization. However in fire engineering applications it may not be possible to process a large number of models concurrently as the results of earlier models often determine parameters for later models.

While FDS model run-time is a critical path component of a fire engineering project it is preceded by model development and followed by analysis and reporting of results. Staged modelling may be more productive as it allows for concurrent model development, analysis, reporting and simulation.

The Pan cluster has the computational capacity to run hundreds of FDS models concurrently, each with optimum parallelization. Concurrent modelling on this platform will provide the fastest processing solution where a large number of simulations are required.

The Beowulf cluster has limited parallel processing resources. While the fastest individual model run-time is achieved by committing all 16 processes to MPI on a single model, the system can be used more efficiently by running two (or more) models in parallel, each with less than the optimum MPI processes for minimum run-time.

Table 4 summarises the overall processing time benefit of concurrent modelling on the Beowulf cluster for the 2M model.

| Number of Concurrent Models | MPI Processes per Model | Total Processing Time (hours) | Improvement over Serial Processing |
|---|---|---|---|
| 1 | 16 | 6.7 | 0% |
| 2 | 8 | 11.2 | 17% |
| 4 | 4 | 19.7 | 26% |
| 8 | 2 | 26.6 | 50% |

Table 4. Concurrent Modelling Efficiency and Run-Time
(Beowulf cluster, 2M model)

## 10.6 Variation in MPI Processing Time

The key measurement from all of the models run for this project was run-time as reported in the FDS .out file. These individual measurements do not provide an understanding the variability of run-time for any particular model. Some variability is expected due to changes in OS demands over time, network availability and processing latency.

In order to examine run-time variability two models were run 100 times on the Beowulf cluster with the resulting data summarised as probability distributions. The task was automated by a shell script file (Appendix C) which extracted and compiled individual model run-times before overwriting the FDS output files.

While it would have been instructive to complete similar tests on the Pan cluster this could not be completed within the allocated computing resources provided for the project.

The two fastest test models from the 32K and 260K computational domains were used in this test, each with 16 MPI processes. The 2M model was not tested because processing time would have been excessive (estimated to be 600 hours).

Summary statistics are presented in Table 5 and the associated probability plots are shown in Figures 14 and 15.

| Computational Domain | Mean Run-Time (s) | Standard Deviation (s) |
|---|---|---|
| 32K | 190.6 | 2.15 |
| 260K | 1,786.9 | 7.69 |

Table 5. Summary Run-Time Statistics

The probability plots show that MPI run-time on the Beowulf cluster is Normally Distributed with little variance. It follows that the individual run-time data measurements are likely to be good estimates of typical run-time performance.

Run-time can be expected to be within +/- 3% of the mean for 99.7% of simulations (3 Sigma limits).

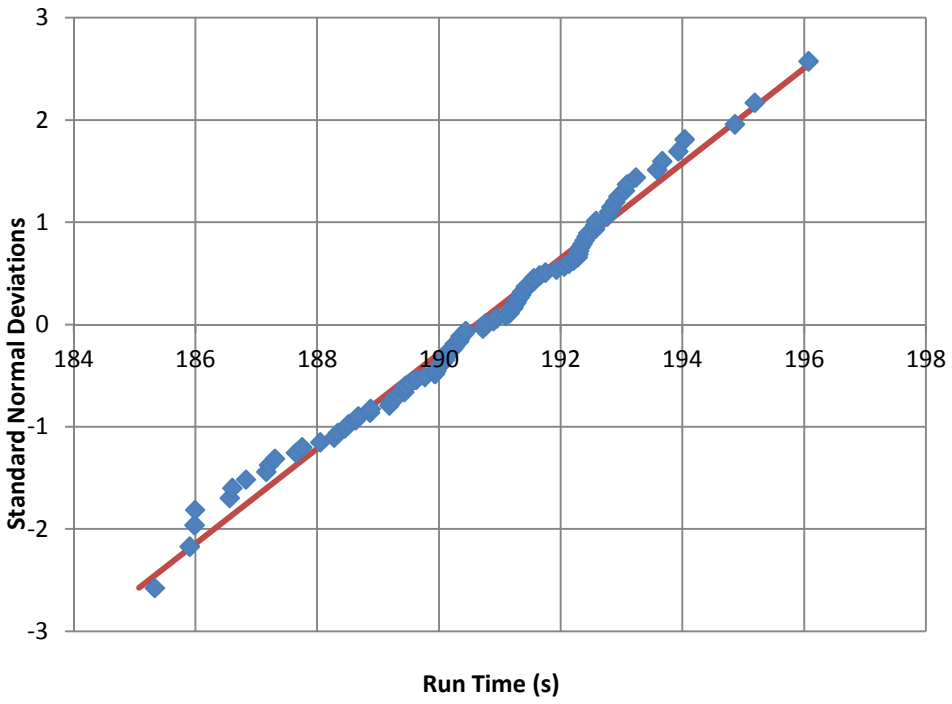


Figure 14. 32K Domain Run-Time Probability Plot

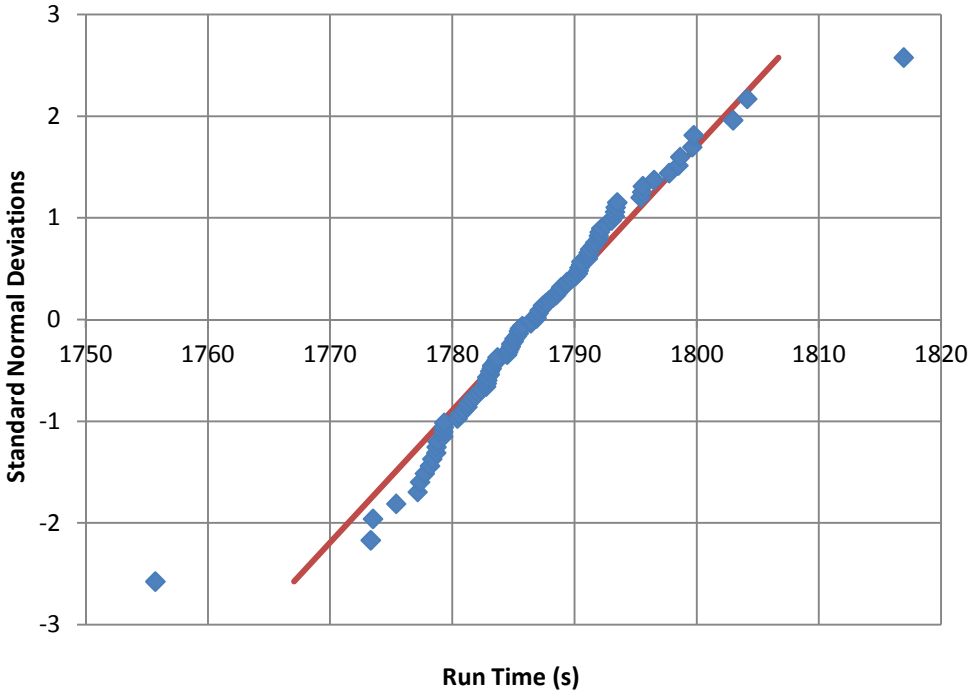


Figure 15. 260K Domain Run-Time Probability Plot

## 10.7 Variation in FDS Output

Ideally, a given model with a given resolution will produce identical FDS outputs on different platforms and/or with different mesh configurations. However mesh boundary effects, the finite resolution of digital computer number systems, differences in numeric algorithms (in firmware or software) and the cumulative effects of rounding over large numbers of calculations are expected to cause variation in FDS output.

Typical FDS DEVICE outputs (a point smoke detector and two thermal LINKS) were used as metrics to compare identical FDS models run on different hardware platforms. Refer to the FDS model at Appendix A for DEVICE characteristics and locations.

The measured data and associated summaries are contained in Appendix E.

Changes in the computational domain cell size are expected to change the output metrics. This is a direct consequence of the spatial averaging that occurs as computational domain cell dimensions are changed. This aspect of model performance would normally be the subject of sensitivity analysis when selecting the computational cell size (Section 4.6 above).

It was also expected that a smaller computational domain cell size should generally produce a 'better' simulation. It follows that DEVICE activation in the 2M model will be a better representation of the performance of a real device than either the 260K or 32K models.

### 10.7.1 FDS Verification

While FDS is subjected to comprehensive testing prior to the release of any update this cannot possibly account for every possible user environment including hardware, operating system and compiler. It follows that an FDS installation on any particular computer platform should be subjected to verification. The reasons for this are explained in detail in the FDS Verification Guide[19] which also provides an installation verification test suite in Appendix B, Table B1. This project is primarily about FDS run-time hence the veracity of the FDS output has not been subject to verification on the Pan Cluster. One might reasonably expect that verification should be completed on every FDS installation, but anecdotally verification is seldom completed.

### 10.7.2 Findings

A given model, platform and parallelization produced identical results for all recorded FDS output variables. This held throughout the hundreds of run-time variation tests described in Section 10.5.

A given model run on different platforms and/or with different parallelization produced changes in the FDS output variables.

On a given hardware platform the 32K metrics were larger (corresponding to delayed activation) compared with the 260K and 2M metrics. This is considered to be a direct consequence of spatial averaging.

Variation of the metrics tended to increase with increasing computational domain size (with the exception on the Pan Cluster 2M models). This is considered to be due to the increased number of calculations required to process a larger computational domain, and the cumulative effects of finite precision arithmetic.

The maximum percentage variation was greater than +/- 5.6% about the of the mid-range on the Pan Cluster with the 260K model.

### 10.7.3 Discussion

A fire engineering design that hinges on just a few percent of DEVICE activation (or some other parameter of interest) for success should be viewed with a certain amount of scepticism and critically assessed for other safety factors in the design and sensitivity. FDS is, after all, a simplified model of the situation under consideration.

Extensive FDS validation18 suggests that an appropriate model can be expected to produce results within useful bounds of a fire experiment. However fire is a stochastic process and even laboratory experiments under tightly controlled conditions can produce unexpected results.

# 11 Conclusions

The following conclusions are drawn from the body of this report. It should be apparent from the body of this report that these are specific to the hardware, the model and the simulation options selected.

## 11.1 Commercial HPC

Commercial HPC is a viable resource for running FDS when a large number of concurrent simulations must be completed in a relatively short time frame to meet project deadlines.

- Data download times can be an appreciable from commercial providers. Transfer of data using physical media (such as an external hard drive) may be warranted.
- The use of HPC resources will not significantly decrease the time to complete a single simulation. HPC cannot reduce the critical path for projects that require models to be developed serially (for example, where the output of earlier models provides input for later models).
- More modest computational platforms can provide better run-time performance than HPC resources for model development and concurrent processing of a limited number of models.
- FDS verification results should be available for any HPC platform (and more generally for any FDS installation).

## 11.2 Parallelization

Parallelization with MPI and OMP will reduce simulation run-time.

- The extent of run-time reduction is both model and hardware specific. In this project optimum use of parallel processing resources produced run-time improvements of over 90%.
- MPI was demonstrated to be more productive than OMP. The maximum run-time gain from OMP was approximately 50%. Allocation of more than about 8 OMP processes may be counter-productive.
- With modest computational platforms improvements in the overall project FDS processing time may be achieved by concurrent modelling with less than optimal parallelization applied to individual models.

## 11.3 Computational Domain

Simulation run-time increases with increasing computational domain size. Halving the basic cell dimension will increase run-time by a factor of about 16. Models should therefore use the minimum mesh refinement consistent with sensitivity analysis of variables to interest for optimum run-time.

Low resolution run times provide a reasonable estimate of higher resolution run times through scaling.

## 11.4 Run-Time Variability

Run-time for any particular hardware platform and model is expected to be Normally distributed with low variance.

## 11.5 FDS Model Output Variability

A given model on a given platform with a particular assignment of parallelization will produce identical FDS outputs with successive simulations.

Increased parallelization and the specification of additional meshes can lead to increases in FDS model output variability. The extent of variability is expected to be more pronounced with increased model resolution.

Subject to the provision of other safety factors, model sensitivity analysis to parallelization may be warranted when the success of a fire engineering design hinges on results within a few percent of failure thresholds.

## 12 References


1. Fire Dynamics Simulator, National Institute of Standards and Technology (NIST), USA Department of Commerce, Version 6.2.0, 2015

2. Yoh, H et al, 'Parallel Computing of Numerical Simulation in Building Fires', Journal of Computers, Vol. 7, No. 11, Nov. 2012

3. http://www.thunderheadeng.com/pyrosim/benchmarks/#runtime

4. Amdahl, G. M., 'Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities', *AFIPS Conference Proc.* (30): 483–485, 1967

5. McGrattan, K., https://github.com/firemodels/fds-smv/wiki/OpenMP-Notes, 2013

6. McGrattan, K, et al., 'Fire Dynamics Simulator Users Guide', National Institute of Standards and Technology (NIST), USA Department of Commerce, Special Publication 1019, 6th Ed., 2015

7. http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html

8. https://en.wikipedia.org/wiki/Prime95

9. Layton, J., 'Using RSH or SSH Raw Ethernet and Cluster Computing Courses', http://www.clustermonkey.net/Beowulf-List/using-rsh-or-ssh-raw-ethernet-and-cluster-computing-courses.html

10. https://github.com/firemodels/fds-smv/wiki/FDS-Release-Notes, FDS 6.3.0 Output

11. https://wiki.auckland.ac.nz/display/CER/NeSI+Pan+Cluster

12. https://www.nesi.org.nz/about-us

13. https://en.wikipedia.org/wiki/Beowulf_cluster

14. https://groups.google.com/forum/#!msg/fds-smv/xEmzcnHHpTI/_MY3s49RVkUJ

15. Ministry of Business Innovation and Employment, 'C/VM2 Verification Method: Framework for Fire Safety Design for New Zealand Building Code Clauses C1-C6 Protection from Fire', Amendment 4, 2014

16. http://slurm.schedmd.com/documentation.html

17. http://www.clustermonkey.net/Beowulf-List/using-rsh-or-ssh-raw-ethernet-and-cluster-computing-courses.html

18. McGrattan, K. et al., 'Fire Dynamics Simulator Technical Reference Guide, Volume 3: Validation', NIST Special Pub. 1018-3, 6th Ed., 18 Nov. 2015

19. McGrattan, K. et al., 'Fire Dynamics Simulator Technical Reference Guide, Volume 2: Verification', NIST Special Pub. 1018-2, 6th Ed., 13 Apr. 2015

## Appendix A FDS Test Model

The following text describes the proposed FDS input file. Mesh allocations and basic cell dimensions are to be adjusted as appropriate to each test case shown in Tables 1 and 2.

```
&HEAD CHID='HPC'/
&TIME T_END=180.0/
&DUMP RENDER_FILE='HPC.ge1', COLUMN_DUMP_LIMIT=.TRUE., DT_PL3D=30.0,
DT_RESTART=30.0, WRITE_XYZ=.TRUE./
&MISC NOISE=.FALSE./

&MESH ID='MESH', IJK=64,128,32, XB=0.0,8.0,0.0,16.0,0.0,4.0/

&REAC ID='CVM2',
      FYI='CVM2 Ver4',
      FUEL='REAC_FUEL',
      C=1.0,
      H=2.0,
      O=0.5,
      CO_YIELD=0.04,
      SOOT_YIELD=0.07,
      HEAT_OF_COMBUSTION=2.0E4/

&PROP ID='Cleary Photoelectric P2',
      QUANTITY='CHAMBER OBSCURATION',
      ACTIVATION_OBSCURATION=9.7,
      ALPHA_E=1.8,
      BETA_E=-0.8,
      ALPHA_C=0.8,
      BETA_C=-0.8/

&PROP ID='Default',
      QUANTITY='SPRINKLER LINK TEMPERATURE',
      ACTIVATION_TEMPERATURE=68.0,
      RTI=135.0,
      C_FACTOR=0.85/

&DEVC ID='SD', PROP_ID='Cleary Photoelectric P2', XYZ=5.0,3.0,3.75/
&DEVC ID='FEDco', QUANTITY='FED', XYZ=4.25,13.0,2.0/
&DEVC ID='FEDco01', QUANTITY='FED', XYZ=4.25,5.0,2.0/
&DEVC ID='FEDrad', QUANTITY='RADIATIVE HEAT FLUX GAS', XYZ=4.25,13.0,2.0,
      ORIENTATION=0.0,0.0,1.0/
&DEVC ID='FEDrad01', QUANTITY='RADIATIVE HEAT FLUX GAS', XYZ=4.25,5.0,2.0,
      ORIENTATION=0.0,0.0,1.0/
&DEVC ID='FEDtemp', QUANTITY='TEMPERATURE', XYZ=4.25,13.0,0.0/
&DEVC ID='FEDtemp01', QUANTITY='TEMPERATURE', XYZ=4.25,5.0,0.0/
&DEVC ID='LAYER->HEIGHT', QUANTITY='LAYER HEIGHT',
      XB=4.25,4.25,13.0,13.0,0.0,4.0/
&DEVC ID='LAYER->LTEMP', QUANTITY='LOWER TEMPERATURE',
      XB=4.25,4.25,13.0,13.0,0.0,4.0/
&DEVC ID='LAYER->UTEMP', QUANTITY='UPPER TEMPERATURE',
      XB=4.25,4.25,13.0,13.0,0.0,4.0/
&DEVC ID='LAYER01->HEIGHT', QUANTITY='LAYER HEIGHT',
      XB=4.25,4.25,5.0,5.0,0.0,4.0/
&DEVC ID='LAYER01->LTEMP', QUANTITY='LOWER TEMPERATURE',
      XB=4.25,4.25,5.0,5.0,0.0,4.0/
```

```
&DEVC ID='LAYER01->UTEMP', QUANTITY='UPPER TEMPERATURE',
      XB=4.25,4.25,5.0,5.0,0.0,4.0/
&DEVC ID='SLINK', PROP_ID='Default', XYZ=1.75,10.0,3.75/
&DEVC ID='SLINK01', PROP_ID='Default', XYZ=5.0,6.75,3.75/

&SURF ID='FIRE1',
      FYI='Fire',
      COLOR='RED',
      HRRPUA=1.333E4,
      TAU_Q=-655.0/

&OBST XB=4.5,5.5,9.25,10.75,0.0,0.5, PERMIT_HOLE=.FALSE.,
      THICKEN=.TRUE.,SURF_IDS='FIRE1','INERT','INERT'/ Obstruction

&VENT SURF_ID='OPEN', XB=6.25,7.0,16.0,16.0,0.0,2.0/ Vent
&VENT SURF_ID='OPEN', XB=8.0,8.0,1.5,2.25,8.882E-16,2.0/ Vent01

&SLCF QUANTITY='TEMPERATURE', PBY=10.0/
&SLCF QUANTITY='TEMPERATURE', PBX=5.0/
&SLCF QUANTITY='TEMPERATURE', PBZ=2.0/
&SLCF QUANTITY='VISIBILITY', PBY=10.0/
&SLCF QUANTITY='VISIBILITY', PBX=5.0/
&SLCF QUANTITY='VISIBILITY', PBZ=2.0/
&SLCF QUANTITY='VELOCITY', VECTOR=.TRUE., PBY=10.0/
&SLCF QUANTITY='VELOCITY', VECTOR=.TRUE., PBX=5.0/
&SLCF QUANTITY='VELOCITY', VECTOR=.TRUE., PBZ=2.0/
&SLCF QUANTITY='TEMPERATURE', PBX=4.75/
&SLCF QUANTITY='TEMPERATURE', PBY=9.5/
&SLCF QUANTITY='VISIBILITY', PBX=4.75/
&SLCF QUANTITY='VISIBILITY', PBY=9.5/
&SLCF QUANTITY='VELOCITY', VECTOR=.TRUE., PBX=4.75/
&SLCF QUANTITY='VELOCITY', VECTOR=.TRUE., PBY=9.5/
&SLCF QUANTITY='TEMPERATURE', PBZ=3.75/
&SLCF QUANTITY='VELOCITY', PBZ=3.75/
&SLCF QUANTITY='VISIBILITY', PBZ=3.75/
&SLCF QUANTITY='TURBULENCE RESOLUTION', PBX=5.0/
&SLCF QUANTITY='TURBULENCE RESOLUTION', PBY=10.0/
&SLCF PBY=10, QUANTITY='SCALAR RESOLUTION', QUANTITY2='MASS FRACTION',
      SPEC_ID='NITROGEN' /
&SLCF PBX=5, QUANTITY='SCALAR RESOLUTION', QUANTITY2='MASS FRACTION',
      SPEC_ID='NITROGEN' /

&TAIL /
```

## Appendix B SLURM Script

The following is a typical SLURM script used for submission of FDS jobs onto the Pan cluster. The purpose of incorporating this information in this report is to provide the reader with a shell for any FDS submissions that they might make to the Pan cluster, and also to advise that of the need to become familiar with any batch management system on an HPC computing platform.

The technical support personnel and system documentation were exceedingly helpful in facilitating this project and getting FDS to run on the Pan cluster.

The file is modified for the specific FDS input file and the allocation of parallel resources, saved in the model directory with an appropriate file-name with a .sl (script language) extension.

The file is then submitted to SLURM with the command sbatch file-name.sl.

```
#!/bin/bash
#SBATCH -J M05-4-2
#SBATCH -A nesi00202
#SBATCH -p merit
#SBATCH --workdir=/projects/nesi00202/HPC/M05-4-2
#SBATCH --time=100:00:00
#SBATCH -o hpc1_%J.out
#SBATCH -e hpc1_%J.err
#SBATCH --mem-per-cpu=4096
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=2
#SBATCH --mail-type=ALL
#SBATCH --mail-user=xxx@yyy.zzz
#SBATCH -C sb
##########################################

module load FDS/6.2.0-intel-2015a

srun fds_mpi HPC05-4.fds
```

## Appendix C BASH Script

The following is the Linux BASH script used on the Beowulf cluster to complete multiple submissions of a single FDS model with extraction and compilation of run-time and device data for statistical analysis. The output from individual models is progressively overwritten. Note that the \ character for continuation lines has been incorporated in the listing below for clarity.

This is almost certainly not the most elegant script solution ever written (a script file expert is unlikely to be impressed) but it is reasonably self-explanatory, it works, and it is easily modified for extracting other model information.

The script file was named runtime.sh, loaded in the model directory and executed using the file name: runtime.

```
#!/bin/sh

echo "List of Files:"

for cases in {1..100}

do
      mpirun -x OMP_NUM_THREADS=1 -np 16 -host \
      Master,Slave0,Slave1,Slave2 fds HPC20-16.fds

      wait

      time=$(grep -w 'Total Elapsed' \
      HPC20-16.out | grep -o -E '[0-9.]'+)

      sd=$(grep -w '1  SD' HPC20-16.out)
      sd=${sd##*SD}
      sd=${sd%% s}

      as1=$(grep -w '14  SLINK' HPC20-16.out)
      as1=${as1##*SLINK}
      as1=${as1%% s}

      as2=$(grep -w '15  SLINK01' HPC20-16.out)
      as2=${as2##*SLINK01}
      as2=${as2%% s}

      echo $time $sd $as1 $as2 >> runtime.txt

      echo $cases $time $sd $as1 $as2

done
```

## Appendix D Comparative MPI and OMP Graphics

Figures D1 to D11 show the test model run-time difference between the Pan and Beowulf hardware platforms with both OMP and MPI parallelization and increasing mesh resolution.

Analysis of these results is incorporated in Section 10.3.

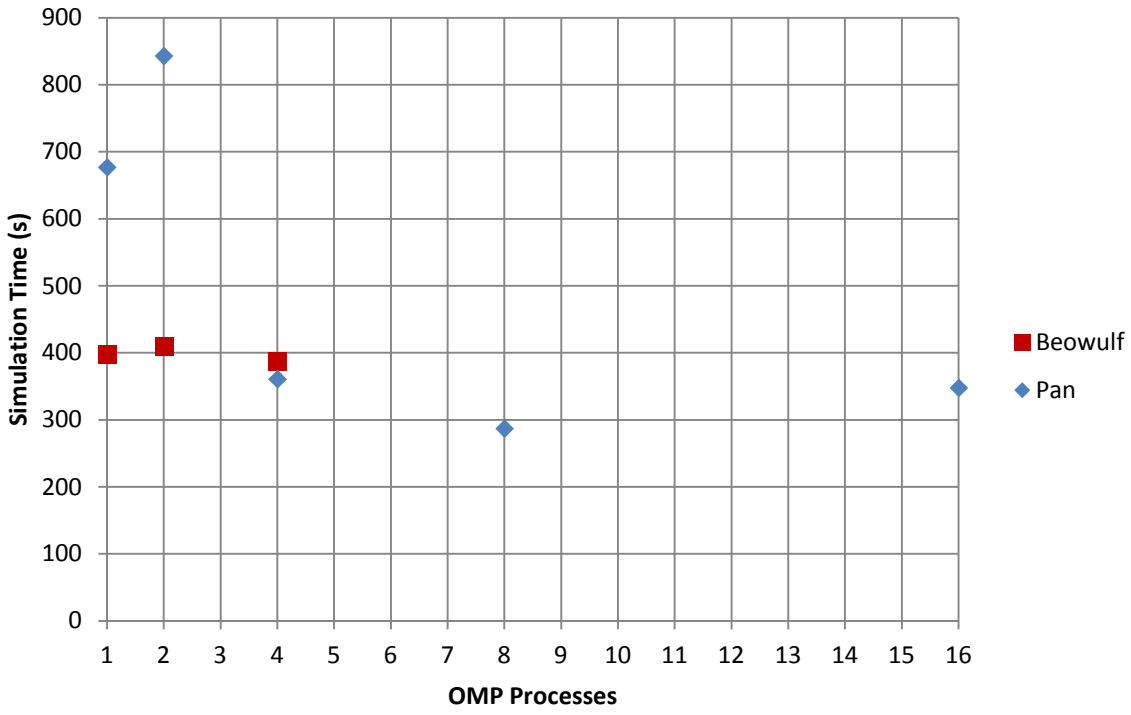


Figure D1. Simulation Time with Increasing OMP Parallelization
2 MPI Processes, 32K Computational Domain

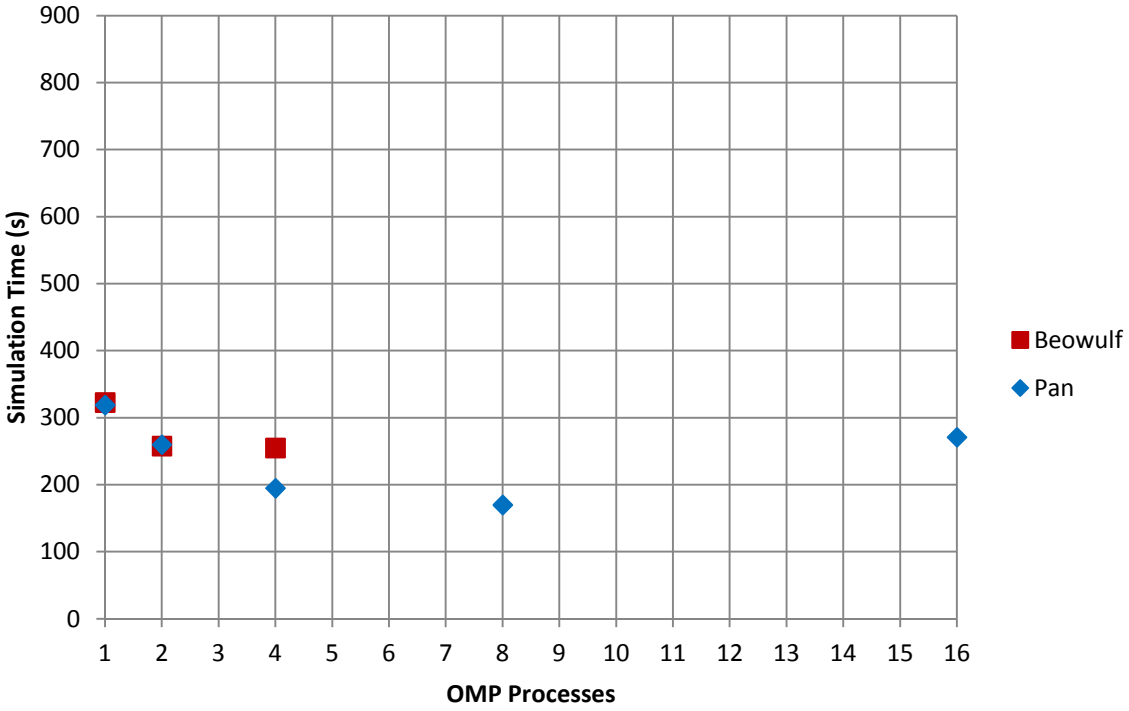


Figure D2. Simulation Time with Increasing OMP Parallelization
4 MPI Processes, 32K Computational Domain

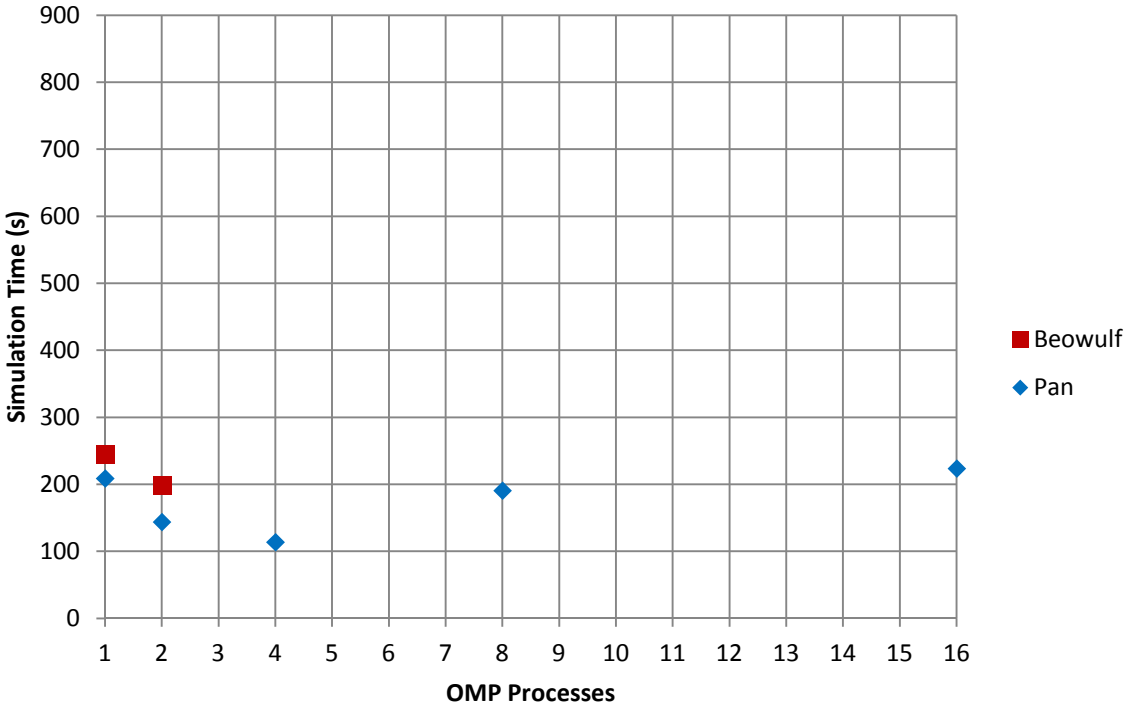


Figure D3. Simulation Time with Increasing OMP Parallelization
8 MPI Processes, 32K Computational Domain

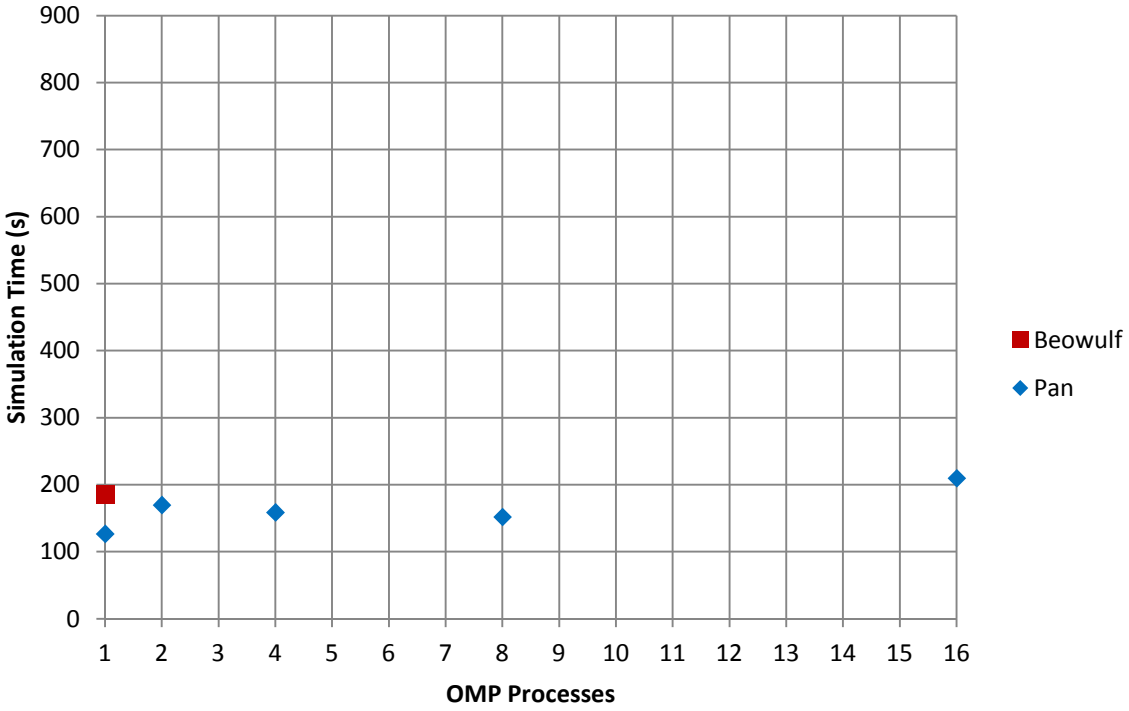


Figure D4. Simulation Time with Increasing OMP Parallelization
16 MPI Processes, 32K Computational Domain

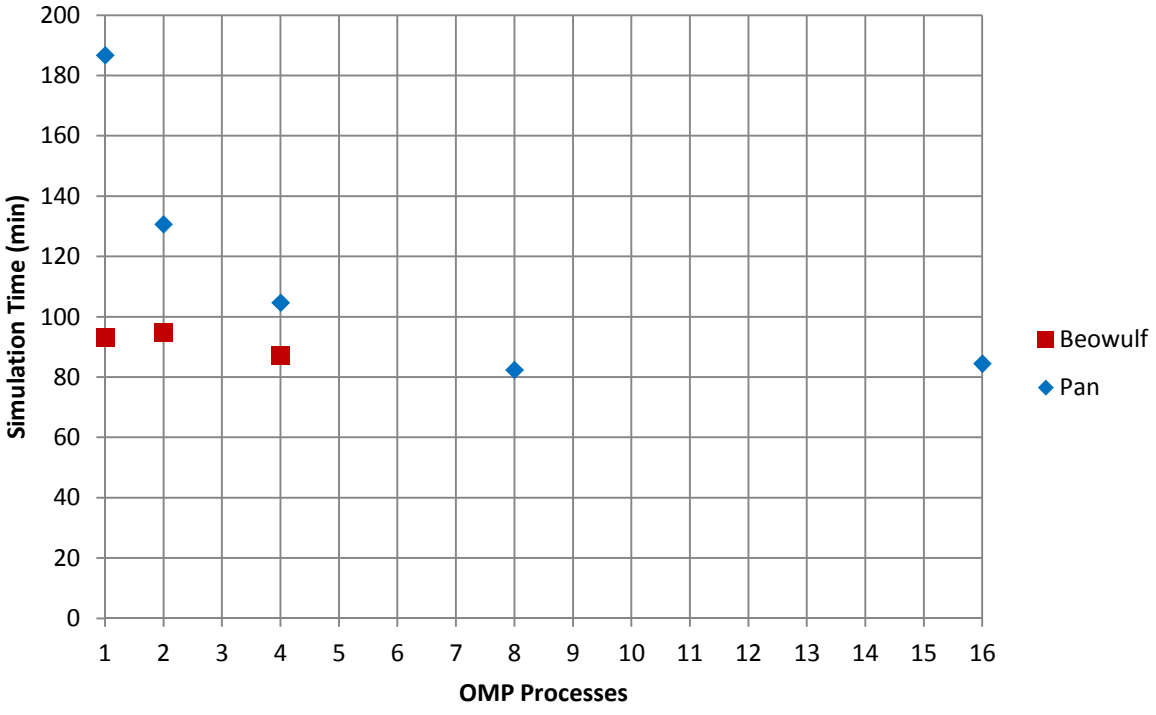


Figure D5. Simulation Time with Increasing OMP Parallelization
2 MPI Processes, 260K Computational Domain

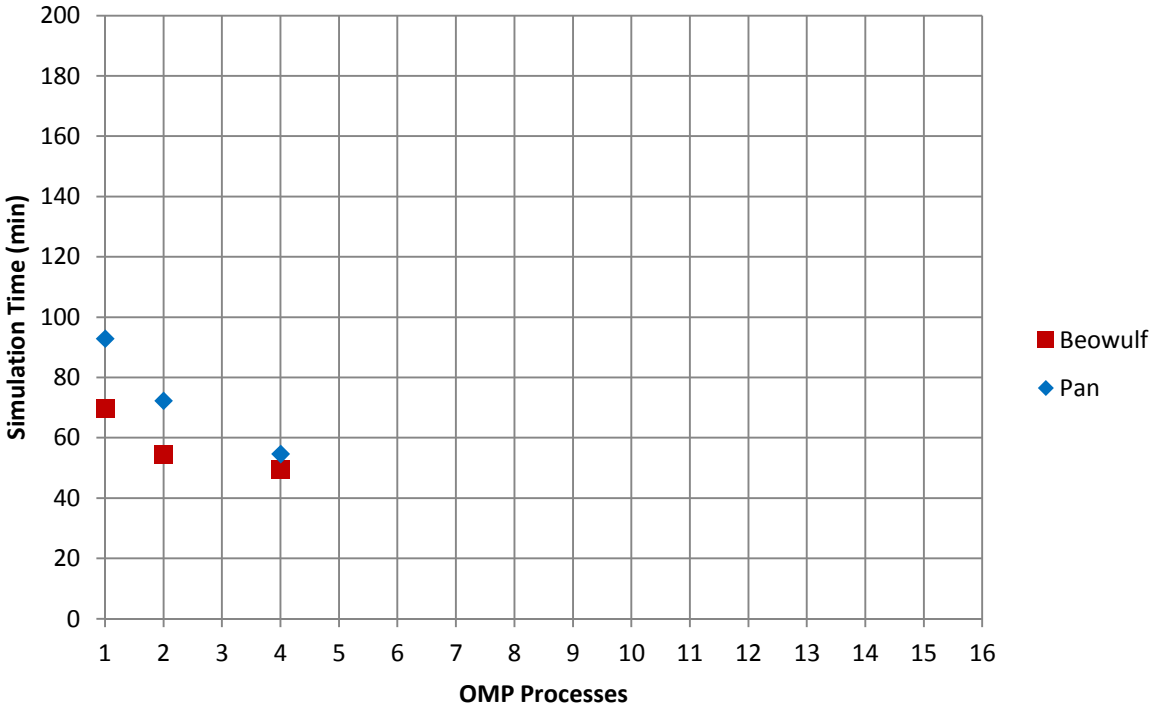


Figure D6. Simulation Time with Increasing OMP Parallelization
4 MPI Processes, 260K Computational Domain

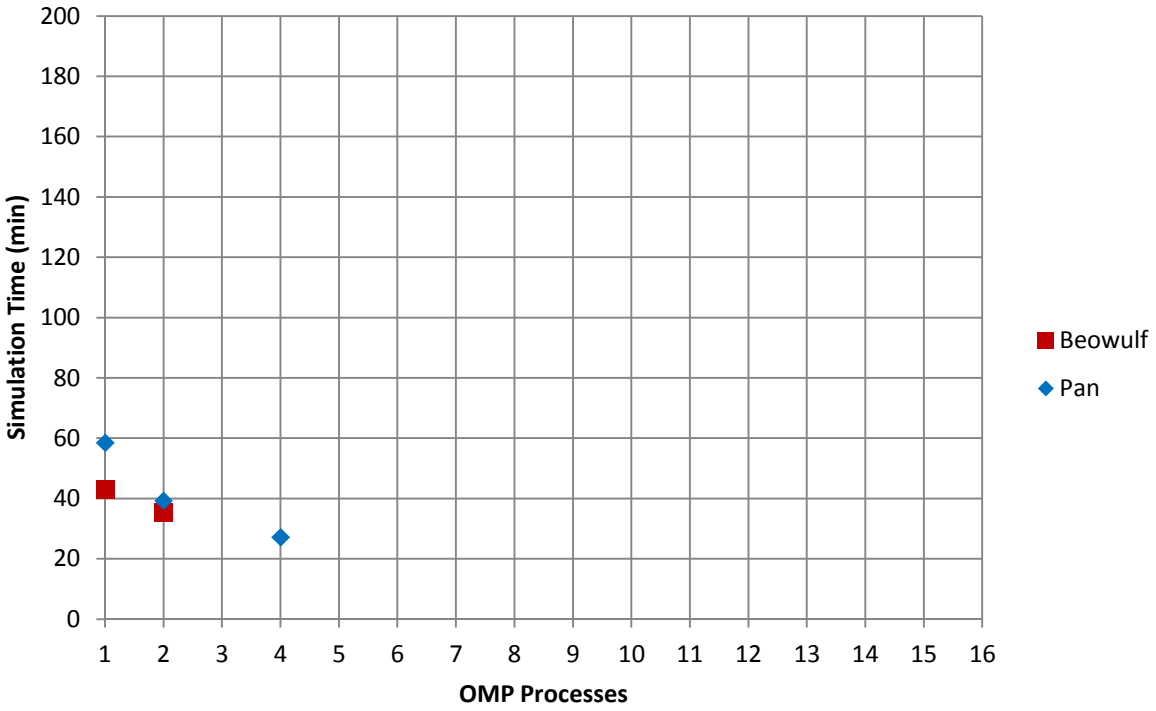


Figure D7. Simulation Time with Increasing OMP Parallelization
8 MPI Processes, 262,144 Cell Computational Domain

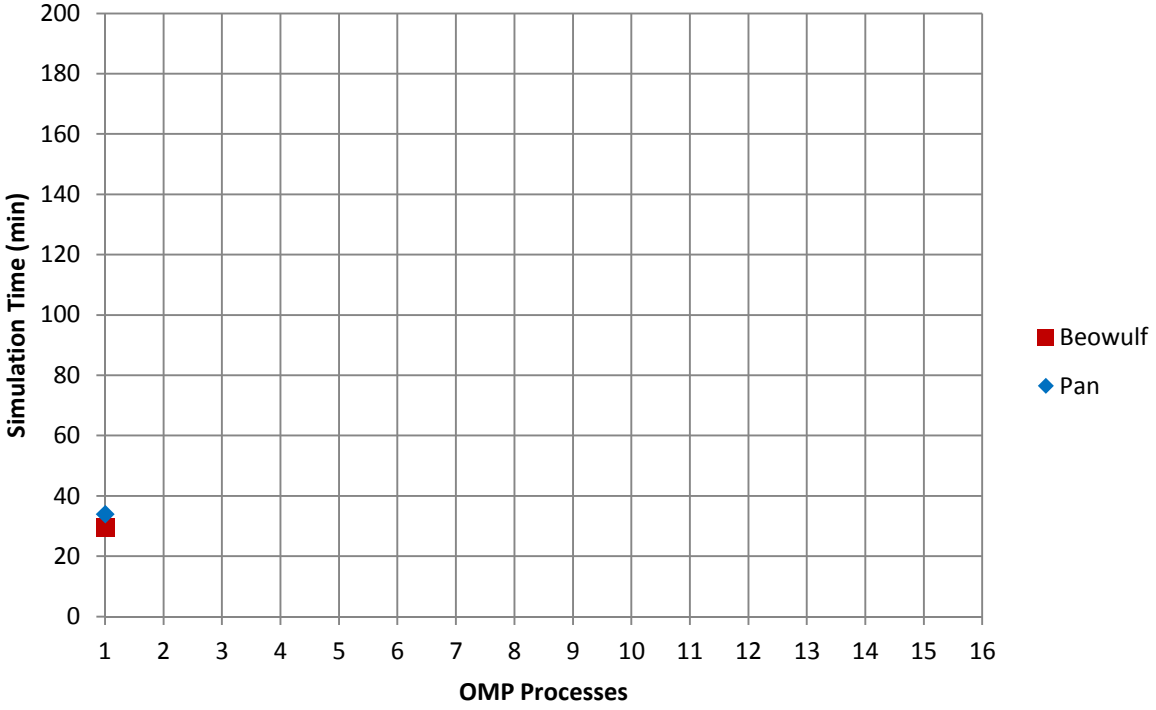


Figure D8. Simulation Time with Increasing OMP Parallelization
16 MPI Processes, 260K Computational Domain

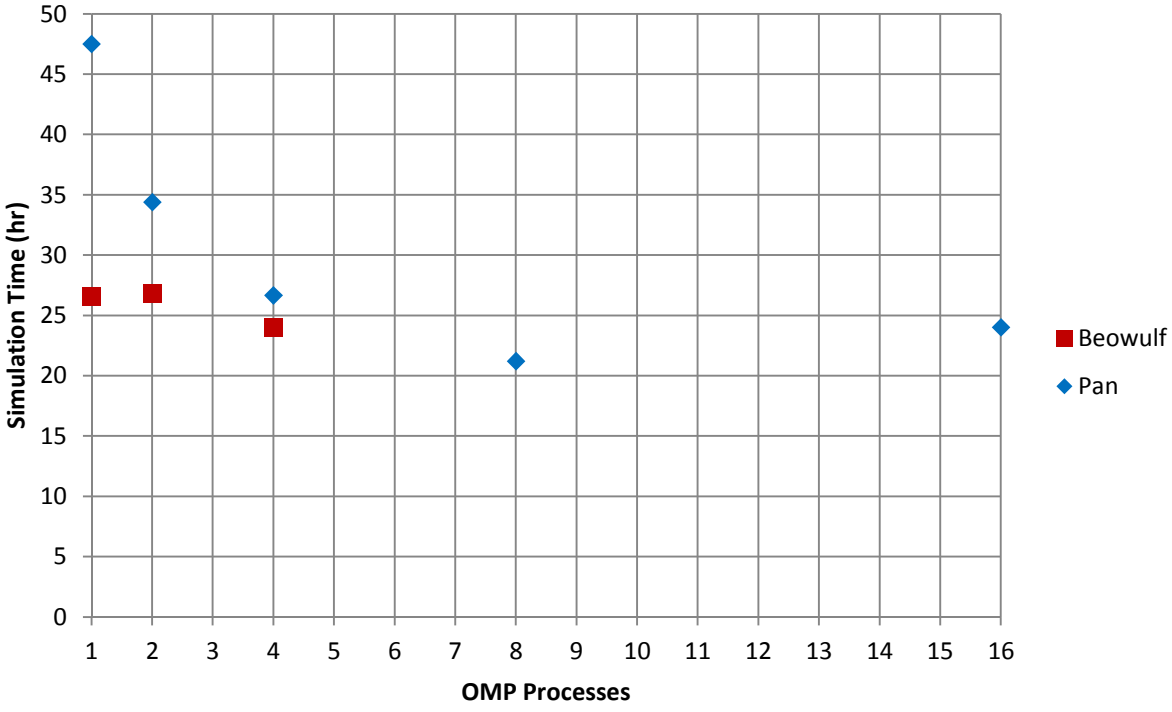


Figure D9. Simulation Time with Increasing OMP Parallelization
2 MPI Processes, 2M Computational Domain

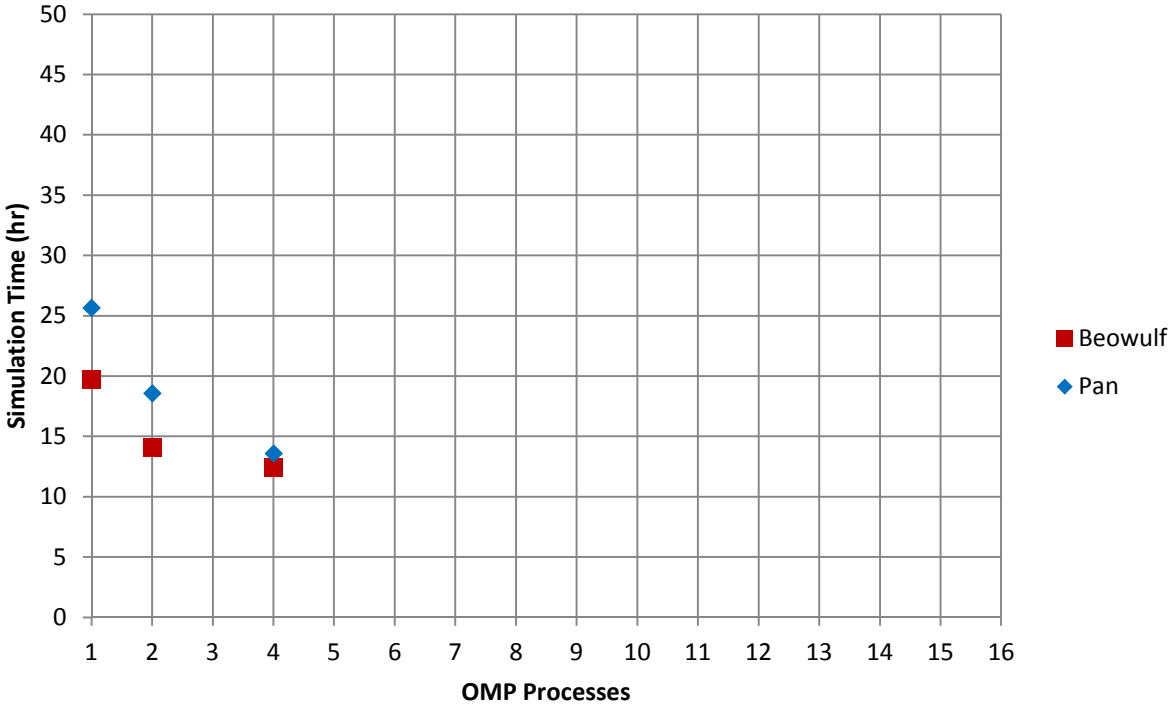


Figure D10. Simulation Time with Increasing OMP Parallelization
4 MPI Processes, 2M Computational Domain

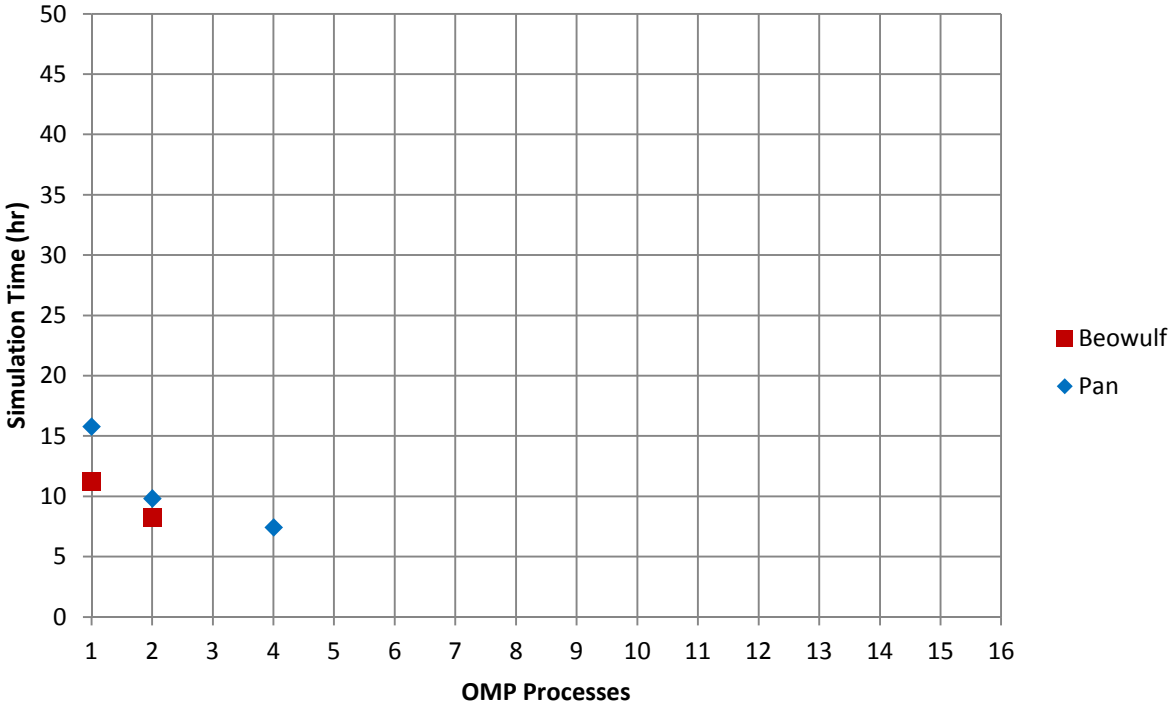


Figure D11. Simulation Time with Increasing OMP Parallelization
8 MPI Processes, 2M Computational Domain

## Appendix E FDS DEVICE Activation Times

| 32K Computational Domain | | | |
|---|---|---|---|
| OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 40.9 | 179.1 | 170.6 |
| 2 | 40.9 | 179.1 | 170.6 |
| 4 | 40.9 | 179.6 | 169.0 |
| | | | |
| Minimum | 40.9 | 179.1 | 169.0 |
| Maximum | 40.9 | 179.6 | 170.6 |
| Range | 0 | 0.5 | 1.6 |
| Mid-Range Variation | +/- 0% | +/- 0.1% | +/- 0.5% |

Table E1. FDS Device Activation Times on Windows Workstation (32K Computational Domain)

| 260K Computational Domain | | | |
|---|---|---|---|
| OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 37.7 | 158.9 | 151.1 |
| 2 | 37.7 | 160.6 | 153.2 |
| 4 | 37.7 | 158.9 | 151.5 |
| | | | |
| Minimum | 37.7 | 158.9 | 151.1 |
| Maximum | 37.7 | 160.6 | 153.2 |
| Range | 0 | 1.7 | 2.1 |
| Mid-Range Variation | +/- 0% | +/- 0.5% | +/- 0.7% |

Table E2. FDS Device Activation Times on Windows Workstation (260K Computational Domain)

| 2M Computational Domain | | | |
|---|---|---|---|
| OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 39.4 | 165.8 | 157.2 |
| 2 | 39.4 | 165.8 | 157.2 |
| 4 | 39.4 | 164.0 | 157.5 |
| | | | |
| Minimum | 39.4 | 164.0 | 157.2 |
| Maximum | 39.4 | 165.8 | 157.5 |
| Range | 0 | 1.8 | 0.3 |
| Mid-Range Variation | +/- 0% | +/- 0.6% | +/- 0.1% |

Table E3. FDS Device Activation Times on Windows Workstation (2M Computational Domain)

| 32K Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 40.9 | 179.1 | 170.6 |
| 1 | 2 | 40.9 | 179.1 | 170.6 |
| 1 | 4 | 40.9 | 179.1 | 170.6 |
| 2 | 1 | 40.9 | 179.1 | 170.6 |
| 2 | 2 | 40.9 | 179.1 | 170.6 |
| 2 | 4 | 40.9 | 179.1 | 169.1 |
| 4 | 1 | 40.9 | 178.1 | 170.4 |
| 4 | 2 | 40.9 | 178.1 | 170.4 |
| 4 | 4 | 40.9 | 178.1 | 170.4 |
| 8 | 1 | 41.1 | 179.4 | 168.5 |
| 8 | 2 | 41.1 | 179.4 | 168.5 |
| 16 | 1 | 41.1 | 179.4 | 169.8 |
| | | | | |
| Minimum | | 40.9 | 178.1 | 168.5 |
| Maximum | | 41.1 | 179.4 | 170.6 |
| Range | | 0.2 | 1.3 | 2.1 |
| Mid-Range Variation | | +/- 0.3% | +/- 0.4% | +/- 0.6% |

Table E4. FDS Device Activation Times on Beowulf Cluster
(32K Computational Domain)

| 260K Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 37.7 | 158.9 | 151.5 |
| 1 | 2 | 37.7 | 158.9 | 151.5 |
| 1 | 4 | 37.7 | 158.9 | 151.5 |
| 2 | 1 | 38.0 | 158.1 | 150.3 |
| 2 | 2 | 38.0 | 158.1 | 150.3 |
| 2 | 4 | 38.0 | 158.1 | 150.3 |
| 4 | 1 | 37.6 | 159.2 | 153.7 |
| 4 | 2 | 37.6 | 159.2 | 153.7 |
| 4 | 4 | 37.6 | 159.2 | 153.7 |
| 8 | 1 | 38.9 | 158.6 | 151.3 |
| 8 | 2 | 38.9 | 158.6 | 151.3 |
| 16 | 1 | 37.3 | 157.6 | 153.1 |
| | | | | |
| Minimum | | 37.3 | 157.6 | 150.3 |
| Maximum | | 38.9 | 159.2 | 153.7 |
| Range | | 1.6 | 1.6 | 3.4 |
| Mid-Range Variation | | +/- 2.1% | +/- 0.5% | +/- 1.1% |

Table E5. FDS Device Activation Times on Beowulf Cluster
(260K Computational Domain)

| 2M Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 39.4 | 165.8 | 157.2 |
| 1 | 2 | 39.4 | 165.8 | 157.2 |
| 1 | 4 | 39.4 | 165.8 | 157.2 |
| 2 | 1 | 37.8 | 163.8 | 157.5 |
| 2 | 2 | 37.8 | 163.8 | 157.5 |
| 2 | 4 | 37.8 | 163.5 | 155.6 |
| 4 | 1 | 39.7 | 163.1 | 157.4 |
| 4 | 2 | 40.9 | 178.1 | 170.4 |
| 4 | 4 | 39.7 | 163.1 | 157.4 |
| 8 | 1 | 37.3 | 166.5 | 156.7 |
| 8 | 2 | 37.3 | 166.5 | 156.7 |
| 16 | 1 | 38.9 | 162.4 | 158.5 |
| | | | | |
| Minimum | | 37.3 | 162.4 | 155.6 |
| Maximum | | 40.9 | 178.1 | 170.4 |
| Range | | 3.6 | 15.7 | 14.8 |
| Mid-Range Variation | | +/- 4.6% | +/- 4.6% | +/- 4.5% |

Table E6. FDS Device Activation Times on Beowulf Cluster
(2M Computational Domain)

| 32K Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 41.0 | 179.8 | 169.5 |
| 1 | 2 | 41.0 | 179.8 | 169.5 |
| 1 | 4 | 41.0 | 179.8 | 169.5 |
| 1 | 8 | 41.0 | 179.8 | 169.5 |
| 1 | 16 | 41.0 | DNA | 170.2 |
| 2 | 1 | 40.9 | 178.5 | 169.4 |
| 2 | 2 | 40.9 | 178.5 | 169.4 |
| 2 | 4 | 40.9 | 178.4 | 169.4 |
| 2 | 8 | 40.9 | 178.5 | 169.4 |
| 2 | 16 | 40.9 | 178.8 | 170.3 |
| 4 | 1 | 40.9 | 179.1 | 170.2 |
| 4 | 2 | 40.9 | DNA | 170.8 |
| 4 | 4 | 40.9 | DNA | 170.1 |
| 4 | 8 | 40.9 | 178.9 | 171.5 |
| 4 | 16 | 40.9 | 179.0 | 170.3 |
| 8 | 1 | 41.1 | 179.7 | 170.2 |
| 8 | 2 | 41.1 | 179.6 | 170.3 |
| 8 | 4 | 41.1 | 180.0 | 170.2 |
| 16 | 1 | 41.2 | 178.9 | 170.5 |
| 16 | 2 | 41.2 | 179.9 | 170.1 |
| 16 | 4 | 41.2 | 178.9 | 170.5 |
| 16 | 8 | 41.2 | 178.9 | 170.5 |
| 16 | 16 | 41.2 | 179.2 | 170.2 |
| | | | | |
| Minimum | | 40.9 | 178.4 | 169.4 |
| Maximum | | 41.2 | >180.0 | 171.5 |
| Range | | 0.3 | >1.6 | 2.1 |
| Mid-Range Variation | | +/- 0.4% | >+/-0.5% | +/- 0.6% |

Note: DNA: Did Not Activate within the 180 second simulation time.

Table E7. FDS Device Activation Times on Pan Cluster
(32K Computational Domain)

| 260K Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 37.8 | 162.6 | 155.5 |
| 1 | 2 | 37.8 | 162.6 | 155.5 |
| 1 | 4 | 37.8 | 162.6 | 155.5 |
| 1 | 8 | 37.8 | 162.6 | 155.5 |
| 1 | 16 | 37.8 | 160.8 | 154.4 |
| 2 | 1 | 38.1 | 162.9 | 156.8 |
| 2 | 2 | 38.1 | 162.9 | 156.8 |
| 2 | 4 | 38.1 | 164.2 | 157.1 |
| 2 | 8 | 38.1 | 162.9 | 156.8 |
| 2 | 16 | 38.1 | 163.5 | 154.6 |
| 4 | 1 | 37.7 | 161.6 | 158.6 |
| 4 | 2 | 37.7 | 162.6 | 155.9 |
| 4 | 4 | 37.7 | 161.7 | 157.4 |
| 8 | 1 | 38.8 | 162.8 | 156.3 |
| 8 | 2 | 38.8 | 162.8 | 156.3 |
| 8 | 4 | 38.8 | 162.2 | 157.2 |
| 8 | 8 | 41.1 | 179.8 | 170.2 |
| 8 | 16 | 41.1 | DNA | 170.6 |
| 16 | 1 | 37.2 | 162.0 | 156.5 |
| | | | | |
| Minimum | | 37.2 | 160.8 | 154.4 |
| Maximum | | 41.1 | >180.0 | 170.6 |
| Range | | 3.9 | >19.2 | 16.2 |
| Mid-Range Variation | | +/- 5.0% | >+/-5.6% | +/- 5.0% |

Note: DNA: Did Not Activate within the 180 second simulation time.

Table E8. FDS Device Activation Times on Pan Cluster
(260K Computational Domain)

| 2M Computational Domain | | | | |
|---|---|---|---|---|
| MPI | OMP | Smoke (s) | Link (s) | Link01 (s) |
| 1 | 1 | 38.3 | 168.2 | 160.8 |
| 1 | 2 | 38.3 | 167.5 | 159.1 |
| 1 | 4 | 38.3 | 167.7 | 159.7 |
| 1 | 8 | 38.3 | 168.0 | 159.6 |
| 1 | 16 | 38.3 | 167.5 | 160.0 |
| 2 | 1 | 39.4 | 168.3 | 160.5 |
| 2 | 2 | 39.4 | 167.0 | 160.0 |
| 2 | 4 | 39.4 | 167.3 | 160.3 |
| 2 | 8 | 39.4 | 166.6 | 160.5 |
| 2 | 16 | 39.4 | 167.5 | 159.5 |
| 4 | 1 | 37.2 | 168.0 | 159.6 |
| 4 | 2 | 37.2 | 167.4 | 159.4 |
| 4 | 4 | 37.2 | 169.2 | 160.3 |
| 8 | 1 | 37.8 | 166.3 | 162.1 |
| 8 | 2 | 37.8 | 167.5 | 160.5 |
| 8 | 4 | 37.8 | 167.1 | 159.5 |
| 16 | 1 | 38.7 | 169.5 | 159.2 |
| | | | | |
| Minimum | | 37.2 | 166.3 | 159.1 |
| Maximum | | 39.4 | 169.5 | 162.1 |
| Range | | 2.2 | 3.2 | 3.0 |
| Mid-Range Variation | | +/- 2.9% | +/-1.0% | +/- 0.9% |

Table E9. FDS Device Activation Times on Pan Cluster
(2M Computational Domain)