Competition Proposal for ASC

(Shared by Nanyang Technological University)

9.1 Introduction

The ASC student cluster preliminary competition requires a proposal submission as a prequisite for evaluation by the competition judges for finalist entry selection. This chapter will provide an insight to the effort and suggested methods required for preparing and guiding a team to overcome these challenges on a first attempt. The proposal demonstrates the student's theoretical ability on cluster competition subjects ranging from hardware design/architecture, parallel programming, software applications and code optimisation. This preliminary phase also provides an excellent chance to teach practical aspects of High Performance Computing (HPC) or supercomputing in a very competitive learning environment.

During the training phases, aspiring students learn valuable skills that will help them in their future working careers, team skills, planning, media exposure and technical writing skills. They also have the opportunity of interacting with experts during the various courses conducted and consultation sessions from application subject matter experts.

This guide serves to provide and prepare future aspiring participants on technical writing guidelines, the best practise and strategies to prepare and submit a professional document. The road to the finals begins with this initial task of writing this competition proposal which will bring about personal fulfilment during your undergraduate life experience. The following sections provide the necessary details on how to prepare and achieve this target.

9.2 Team Selection and Composition

The primary task for interested students is to form a team with various skill sets comprising computer engineering, computer science and students from the Sciences/Engineering faculties to undertake the various challenges outlined in the proposal. (However this team combination may not be possible, the reasons will not be discussed in this chapter.)

The interest to participate in ASC competition will require substantially more effort to prepare as compared to International Supercomputing Conference (ISC) and Supercomputing Conference (SC) competitions as the questions require a practise oriented/hands-on approach. Nonetheless the effort to prepare the proposal provides future student talent an insight to the world of supercomputing from an entry level perspective. The proposal preparation process will be facilitated by the team advisor and his/her staff to ensure a team with different skills available for final submission to the competition committee for judging.

9.3 Basic Requisite

It may seem difficult at first glance to prepare for this daunting task. The required tasks and guidance shall provide an overview for aspiring participants and stakeholders to overcome this challenge. As a basic requirement, having Linux fundamentals, some understanding and exposure to handling of servers would be highly beneficial. Students may have their first exposure during their academic studies or through their internships in the industry (if applicable). This is very important as the application section requires code compilation and this involves an in depth understanding. Additional training involving the relationship of the software layer should be introduced as part of the program to provide a complete overview of the basic core concepts.

It is recommended that participation for such competitions receives the university's management support and a team of dedicated staff appointed to support and train the students. Note that this may also be possible with the students spearheading the effort to self-learn and prepare the proposal. However this is highly dependent on the student's initiative, resourcefulness and team chemistry.

9.4 Training

Training forms an essential component and foundation to provide participants the basic core concepts to undertake the competition proposal. As the timeline is rather short and spread over Christmas, New Year's day and the Chinese Spring festival, an extremely well planned training plan is mandatory. A well developed training

program should be established to cover subjects ranging from computing theory, parallel programming, computer hardware, mathematics and software applications. As part of the training curriculum, short work assignments can be incorporated to complement understanding and sustain interest. Students who are from Engineering and Science faculties will initially face a daunting and uphill task to understand and appreciate the subjects. However it is best to leverage on their expertise in the application software whilst progressively equipping them with computer engineering/science concepts.

9.4.1. Training Plan

The training can be developed in 2 phases, basic theory during the initial stages prior to the release of the preliminary competition details and requirements. Such trainings will usually be classroom based. Second phase training can be targeted more specifically towards working on the proposal requirements. This includes hands on practise. (Hint: Participants may refer to the previous year's competition proposal requirements for an overview of the expected software application, usually hardware design, Linpack test, Scientific/Engineering application and code optimisation.)

As most students may/do not have any prior knowledge to work within a supercomputing environment, it is imperative that the training materials are written in simple language and terms to facilitate their understanding, interest and aid their progressive understanding into advanced content.

Providing essential training support through offline consultation is very helpful in providing students the opportunity to clarify their doubts and concerns on a personal basis. (It is recommended the trainer also prepare the subject matter in advance as part of the training effort before providing consultation.)

To ensure a complete range of training content is administered to the students, it is essential to collaborate between your university and a research institute & industry vendor or within the various schools in your university to support this effort. As three are currently three established competitions (SC, ISC and ASC) across the world today, there is considerable information available for prior training preparation.

9.5 Training Hardware

The competition proposal provides a balanced content covering design and practise oriented tasks where participants are required to progressively learn and put their skills and knowledge to use in the form of throughput results generated from the scientific/engineering application runs. It is beneficial to seek out a hardware vendor early and establish a close relationship with them for hardware training sponsorship. Leveraging on existing vendors partnerships can be a great starting point. On the other hand, any additional server equipment from your research centre could also be utilised for hands on practise to setup and establish a mini cluster.

9.5.1. Hardware Platform

There are several types of hardware test platforms available for this purpose: laptops where participants may practise operating system installation and application compilation, a two node cluster for testing and code compilation for distributed codes (MPI). This 2 node cluster is extremely helpful in ensuring that repeatable results can also be replicated on the test platform provided by the ASC competition organiser. It would be ideal if students can make use of this short window to also learn to setup a 2 node cluster as part of the program.

Variation in performance run results are expected when migrating codes across various test platforms therefore it is essential to understand the cluster setup, software stack and interconnect specification and configuration.

9.6 Competition Proposal

The requirements of the proposal will be made available to participants during the formal release of the competition details. Attempting the competition proposal requires a practise oriented approach and understanding of the concepts to fulfil the tasks. The proposal serves as a formal report designed to convey technical information in a clear and easily accessible format. It will be divided into sections which allow different readers to access different levels of information.

9.6.1 Proposal Sections

It is recommended to prepare the proposal with separate sections; main section for

key findings and appendices for supplementary information. Concepts and result analysis trends should be represented in diagrams and graphs respectively as they provide an alternate form of representation to the readers/judges to complement your proposal.

9.6.2 Guidelines for Graphs and Diagrams Presentation

The following section discusses the suggested guidelines for presenting graphs and diagrams to support your findings in a proposal. Graphing is very useful to visualize and describe the relationship between two variables (i.e. speedups vs CPU cores). The independent variable (manipulating variable, i.e. *CPU cores*) is plotted on the x-axis and dependant variable is plotted on the y-axis (responding variable, *wall time*). Both axes of the graphs should also be labelled with both quantity and units. The graph scales should also fill the entire page if possible as shown in Figure 9-1.

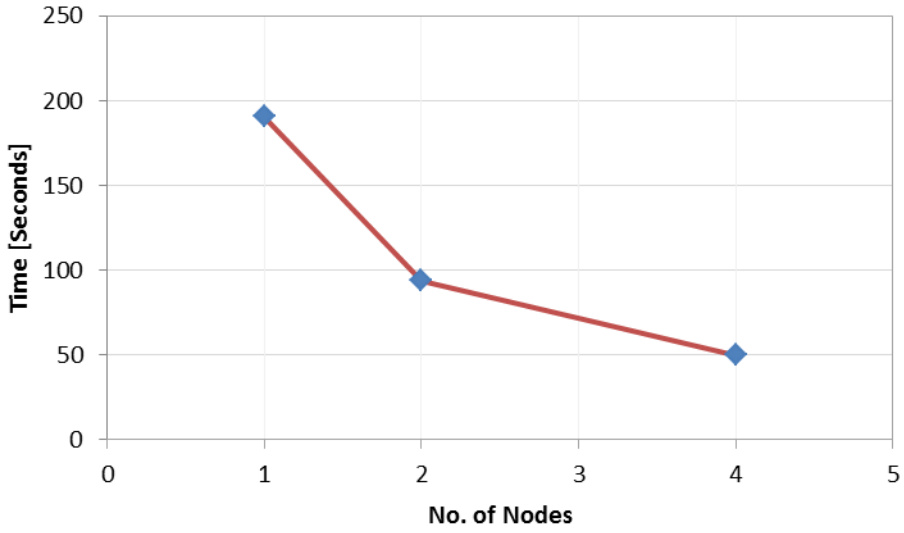

Figure 9-1 Workload Performance Result.

Illustration with diagrams using a suitable font makes the diagram presentable. Objects and fonts are interrelated when it is used to convey a subject/topic. Both the font type and objects in a diagram translate and visualize an idea or concept. Usage of lines in diagrams should be consistent with the types of lines being used. If a dashed line has been used to indicate Ethernet connection, do not use the same line pattern for Infiniband connection in your cluster.

A balance between the objects and whitespace is essential to create a good diagram.

Whitespace can help to emphasize particular elements but also help to balance the objects in the diagram. It must be noted that using blackspace is not highly recommended as it adds a visual strain to the reader's eye and the font colors of the text will not be easily readable. Shadows give the diagram an unclean and artistic feel; therefore it is not recommended for use.

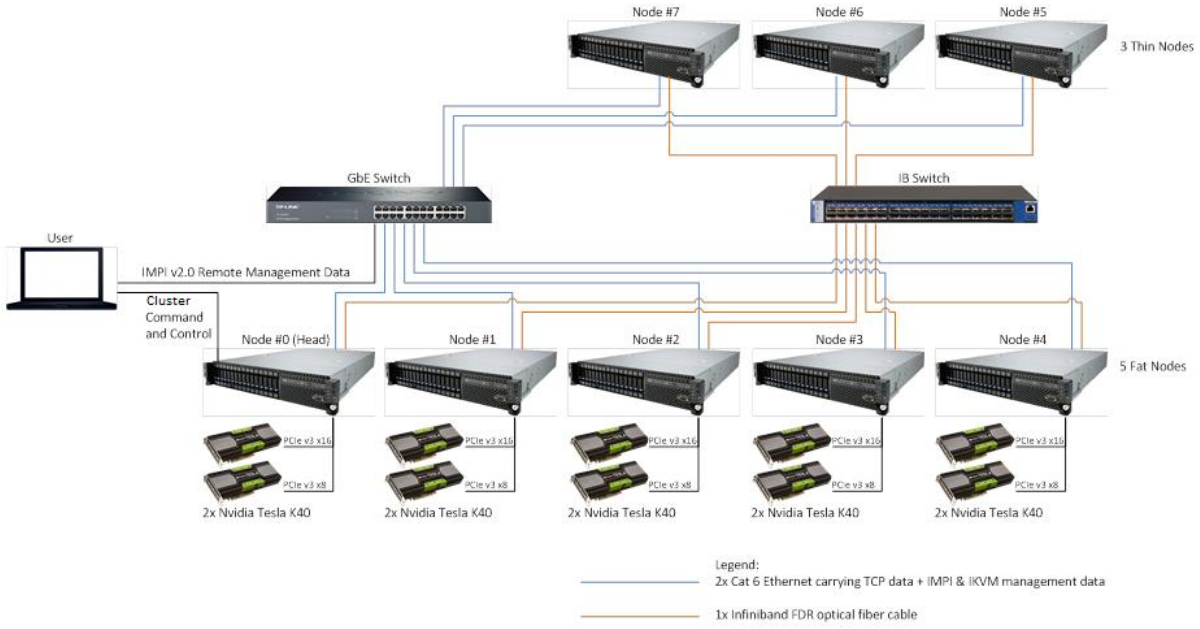

Figure 9-2 Example of Figure with Whitespace and Line Variation.

9.6.3 Guidelines for Other Sections

Acknowledgments may be made to individuals or institutions not mentioned elsewhere in the work who have made significant and important contributions.

9.6.4 Knowledge of High Performance Computing Activities

This section is a relatively simple section which requires you to describe HPC activities in your university or institute. You may provide a description of the hardware and the type of research activities which the system is used for. It will be helpful if you can also provide information of the scale of the massively parallel jobs being executed on the system queue.

9.6.5 Hardware Design and Energy Efficiency

The hardware system design chapter is not directly linked to the domain software

application due to possibility of limited resources across different regions. Therefore you do not need to establish a similar hardware setup to run the jobs based on your theoretical design. In this section, it is important that you understand the importance of having the essential compiler and math libraries installed on your test cluster. Having access to a small cluster will provide participants the chance to verify and implement their findings. These libraries form the basis for ensuring that you can successfully run your Linpack test and quantify your findings.

You may test and run them on any available hardware cluster you have access to. It is also highly recommended to consider discussing the contributions and effects of hardware tuning to operate within the 3kW energy limit. Prior to having access to an operating cluster, tweaking the bios of the server and attaching a power meter to the wall socket provides valuable information on the server's energy consumption performance. Table 0-1 shows the summary of power consumption in watts and the energy efficiency details.

Table 0-1 Test Results with different number of CPU cores.

Cores	**Accelerators**	**Resulting R_{max} [GFlops]**	**Power Consumption [Watts]**	**Energy Efficiency [GFlops/Watt]**
24	1 K40 GPU	1440	700	2.057
18	1 K40 GPU	1364	635 Est.	2.148
12	1 K40 GPU	1253	570	2.198
6	1 K40 GPU	1116	500	2.232
4	1 K40 GPU	1046	475	2.202
2	1 K40 GPU	895	465	1.924

Note: Configuration of HPL follows that of the single node single GPU test

Servers are generally equipped with redundant features (i.e. power supply) for reliable operations in critical business environments. Considerations may also include using energy efficient devices or novel cooling techniques (water cooling) for attaining performance throughput of your cluster. The power usage of powering your miniature water pumps for water cooling shall be covered within the 3kW envelope. (Note: You will sacrifice total CPU cores throughput performance to attain the best Floating Point Operations per Second (FLOPS) in Linpack testing.)

This will serve to demonstrate your understanding and appreciation of hardware

energy efficiency. All final optimised parameters and the necessary accompanying justifications should be duly documented as shown in Table 0-2.

9.6.6 High Performance Linpack (HPL)

High Performance Linpack (HPL) is a computation benchmark which measures the performance of a HPC system by solving a dense matrix of linear equations. In this particular section, understanding the theory and algorithm concept will be helpful in providing the opportunity to evaluate and tune the parameters to achieve your desired results for your cluster. It will be helpful to know that different versions of the HPL code will provide significantly different results. Therefore it is recommended to adopt a consistent version number and libraries across different platforms during your tests.

The concepts and findings from this section of HPL inclusive HPC tuning should be documented and justified to support your selected parameters. Table 0-2 shows an illustration of the summary findings summarised in table format which you may adopt.

Table 0-2 Energy Efficiency of Different Architectures.

Nodes	**CPU cores**	**Accelerator**	**Parallelization Technique**	**N/NB**	**P x Q**	**R_{max} [GFlops]**	**Power [W]**	**Energy Eff. [GFlops/W]**
1	24	None	OpenMP within node	87000/ 224	1x1	473.3	450 ^	1.051
1	24	1 Xeon Phi	OpenMP within node, offload to Xeon Phi	87000/1024	1x1	1185.6	700 ^	1.693
1	24	1 K40 GPU	OpenMP within node, CUDA to GPU	87000/1024	1x1	1440	700 Est.	2.057
2	48	1 **Xeon Phi** per node	OpenMP within node, MPI across nodes, offload to Xeon Phi	123000/1024	1x2	2216.9	1400 Est.	1.583
2	48	1 **K40 GPU** per	OpenMP within node, MPI	123000/1024	1x2	2471	1400 Est.	1.765

		node	across nodes, CUDA to GPU					

CPU: Intel Xeon E5-2695v2: 2.4 GHz * 12 cores * 2 socket * 8 flops/clock = 460.8GFlops per node

MIC: Intel Xeon Phi 5110P: 1.053GHz * 60 cores * 16 flops/clock = 1010.88GFlops per accelerator

GPU: NVidia Tesla K40: According to specifications = 1430GFlops per accelerator

^ Power was measured at the outlet with a power meter

9.6.7 Application Software

The first step in this section is to install the software from source code. There are accompanying instruction files in the source file which you have downloaded. Take a moment to read the contents. Alternatively the developer's website may also contain valuable information and instructions to proceed with your code compilation. (Note: This is valid for all opensource codes) It is easier to start off compiling the software with serial function, test run and ensure a valid output is achieved. This forms a baseline for your verification with distributed runs. Installing the code with parallel capability may require more effort for a first time user and this will improve with more practice.

During the course of optimisation to obtain the best throughput performance, you may compile the code with combinations of different compliers and MPI libraries. This is generally an iterative process but you will gain very much after this challenging endeavour. You may refer to HPC advisory council's website for best practise information on installing various scientific and engineering codes for reference and guidance. You may start off by installing the code by following the suggested options to start off and gain confidence in code compilation.

9.6.8 Document Configuration and Assumptions

In the domain software application, it is recommended to document your understanding of the basic theory and real world application for full appreciation of this subject matter. Benchmarking performance job runs serve to test the understanding of distributed compute runs over serial computation. In fact, understanding the software application requirements (i.e. memory, CPU+accelerator computing capability etc.) is more important than having the best hardware. If the software is not able to leverage on the latest hardware architecture, you will not be able to extract the maximum performance. Edits and amendments to sections of the input file codes should be documented in your proposal to compare with the baseline sample code. Some suggested best practise approach for attempting such a question:

- Installing code with recommended parameters

Based on materials provided on Quantum Espresso webpage, in order to make Quantum Espresso run at its best efficiency, Quantum Espresso must be configured to work with correct settings. The following setup was used as a starting point for running the test shown in Table below.

Table 3: Baseline Setup Values

Configuration	Value
Usage setup	Enabled OpenMP, Enabled Scalapack
Compilers	Compiled with C/C++ – Intel C/C++ (icc), Fortran 77/90 – Intel Fortran (ifort) and Intel MPI (mpiicc and mpiifort)
Libraries	Built with MKL BLAS, LAPACK, SCALAPACK, BLACS and

- Identifying parameters which affect performance of application.

The following parameters have been identified to potentially increase the performance of Quantum Espresso. Specific values used for testing are placed in the parenthesis.

- OpenMP Threads (1, 2, 3, 4)
- Total running processor cores, i.e. OpenMPI cores * OpenMP Threads (12, 24, 48)
- npool
- ndiag

For OpenMP threads, low thread counts as recommended by National Energy Research Scientific Computing Centre. As OpenMP works with a different set of parallelization, compared to MPI. Having cores allocated for MPI-OpenMP hybrid will increase the performance if configured without both techniques conflicting with each other.

The values chosen for total running processor cores is determined by running job on 1 CPU socket, 1 Computing node and finally multiple, i.e. 2 computing node.

- Demonstrating in depth knowledge of configuration related to problem.

Quantum Espresso is written in highly scalable for large parallel computer system, its parallelization has five levels: image, pool, plane-wave, task group and linear-algebra parallelization. However, in two workload, we only use two parallelization options, which is pool (distribute k-points among `npool` of CPUs) and linear-algebra (distribute and parallelize matrix diagonalization and matrix – matrix multiplications needed in iterative diagonalization). The rest of levels were not tweaked because they are not related to our problems, i.e. image and plane-wave parallelization, and the system is not big enough for such scaling to take place.

- Reviewing input file run type and its application.

From the input files given, we can see that workload 1 is about structure relaxation and workload 2 is just a simple self-consistency run.

Judging from the position of atoms, which has been visualized in Figure 9-3, workload 2 is actually dealing with an intrinsic defect, O-vacancy, in a zirconium oxide supercell. Since this kind of intrinsic defects often introduce rich luminescent properties, understanding these defect-related excitions are important to physicists or material scientists and they are particularly important for design and optimization of some nano-materials.

In fact, the calculation was performed in the reciprocal space which deal with the Brilloin-zone with ease, and most often the K-points mesh method is to spread equally spaced in Brilloin-zone, known as Monkhorst and Pack method. In workload 1, 4 4 4 K-points was implemented so along each reciprocal lattice-vectors 4 points was spread there and in workload 2, only centre Γ point was considered.

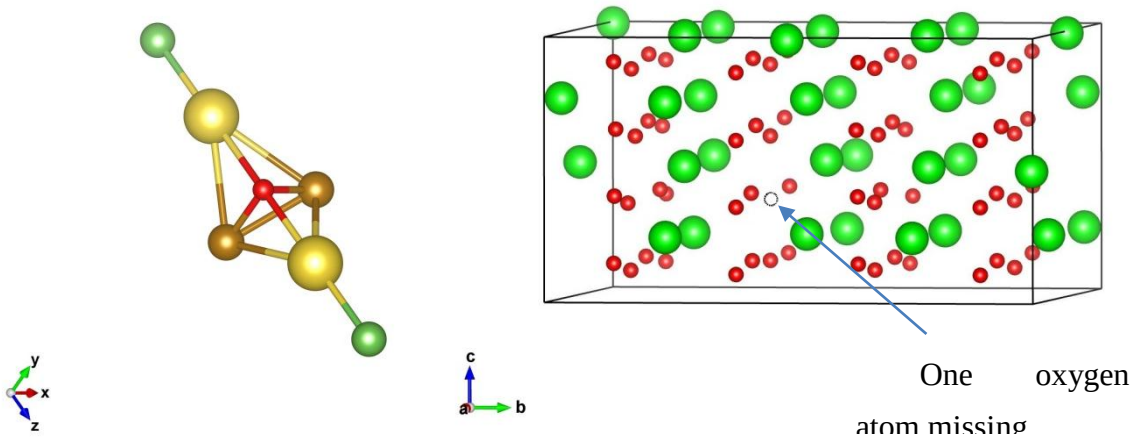

Figure 9-3 Structure of workload 1 and workload 2

In summary, the number of atoms along with parameters encut, nspin, con_thr, kpoints will largely determine the performance and the run time.

- Discussion and review of results.

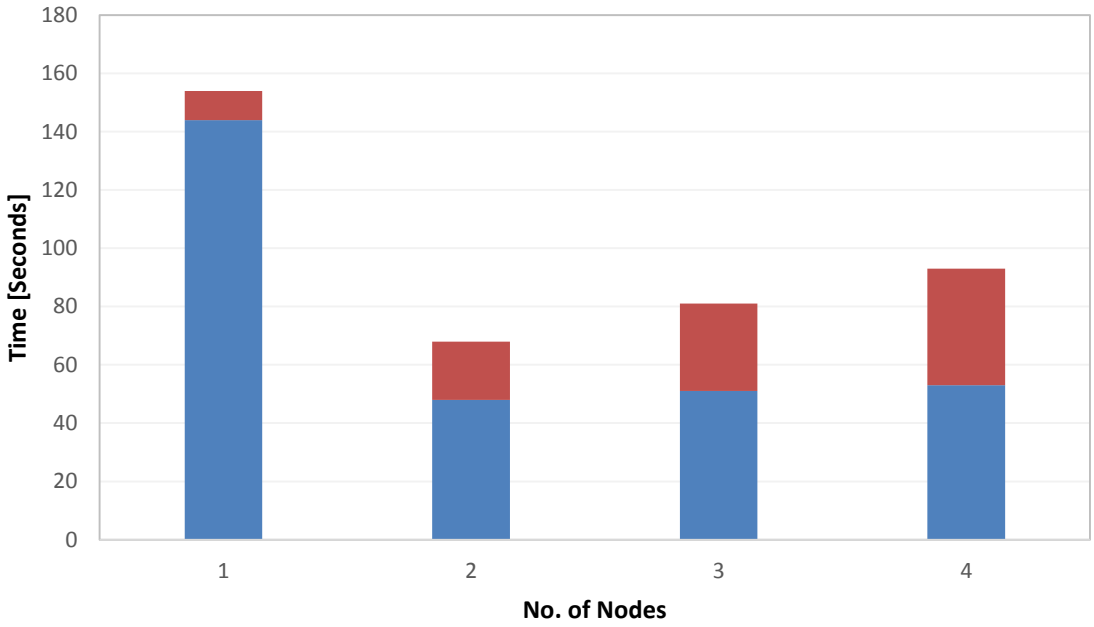

Figure 9-4 Workload 1 npool Comparison.

`npool` is a factor that reduce communication by agglomerating the workload. In the above data, it shows that time decreased from `npool` 1 to `npool` 2; this is due to the amount of nodes the application is running on, i.e. 2 nodes, and thus communication reduces significantly but reducing the need to communicate between nodes frequently. The highlighted orange bar shows a possible overhead of generating the pool, which

shows a linear increment. (10s, 20s, 30s, 40s respectively for increasing `npool`)

The fluctuation of the results from Figure 9-5 was not explainable by any theoretical means. Thus, we came up a few plausible reasons.

1. **Conflicting Hardware Access** – The difference between OpenMP and MPI is that OpenMP threads work on the same set of memories and may result in conflicts when both threads enters heavy data access phases.
2. **Untuned MPI** – During the server setup, Intel MPI Automatic Tuning Utility was not used to check if MPI communication produces consistent speed.

9.6.9 Code Optimisation

For code optimisation, there will be a challenge for both the computer science and application domain (Scientific or Engineering) member to understand the multidisciplinary skills involved. The code optimisation section is assigned the highest score therefore advisor and trainers will need to facilitate and monitor closely to ensure the essential guidance is provided. It must be noted that close work collaboration between team members undertaking this section of the work is required.

The optimisation can be attempted from either programming or rewriting the mathematical algorithm. It must be noted that rewriting the mathematical algorithm requires extensive efforts. The question will indicate whether it is possible to rewrite the maths algorithm and the governing rules on the extent of optimisation.

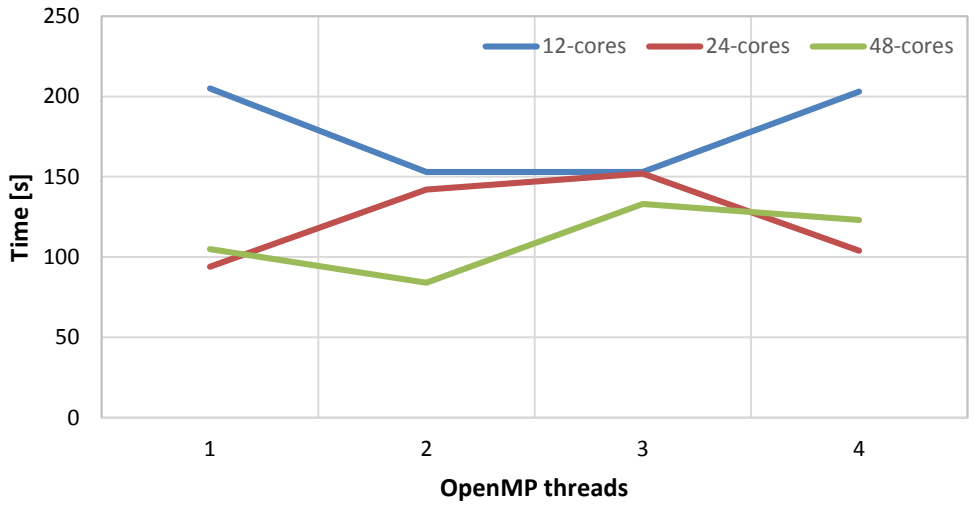

Figure 9-5: Time Taken By Different OpenMP Threads with Different Running

Cores.

Attempt to work on this question requires starting off to evaluate the entire source code. You may like to first run the correctness test to validate its output generated. Good scalability performance of HPC applications involves good understanding of the workload through performing profile analysis, and comparing behaviours of using different type of hardware architecture (i.e. CPU vs CPU+MIC). This will pinpoint bottlenecks in segments of sample code and through this, working on the largest bottleneck can provide you an easier option to tune and improve on the application performance. Thereafter sections of the code which exhibit poor programming techniques can be singled out for improvement.

Next you may describe the various stages of the parallel programming design methodology as an introduction. The next step involves documenting performance optimisation which covers the different processes and the effects of using different compiler flags and runtime parameter configurations have on the code output. It is highly recommended to display code snippets (before and after) to illustrate the improvements to the results in the proposal. The speedup improvements from the various improvements implemented should be documented for comparisons. Reasons/justifications supporting the results will indicate the proficiency level of the team.

9.7 Proposal Task Distribution

It is recommended that the team members split up the tasks amongst the group to work on the various section of the proposal. This approach is likely more efficient and effective as the tasks are executed concurrently.

Amongst the team members, the leader will perform dual roles, a player to work on tasks and leader to spur the team. Close consultation with team advisor and trainers is highly recommended to ensure progress, content is valid and presented in a professionally technical format. Time spent to work on various task varies accordingly to the extent of task effort required. This ranges from 10 – 200 hours spent to attempt each task.

9.8 Summary

Attempting to prepare and submit such a proposal requires a team effort. It will

be an initially daunting task but having a proper training framework in place will be highly effective and beneficial. Having a group of highly motivated individuals will form the basis to establish a team, prepare and submit the proposal. The proposal incorporates a substantial hands-on effort to work on the application software question and code optimisation, therefore having a strong basic foundation of the essential skills and knowledge is very important.

The suggestions and guidelines presented in this chapter will provide first time participants suitable guidance & advice to prepare a plan and prepare the team. Presenting the proposal in a suitable format provides a consistent platform for evaluation. Technical concepts and findings should be correctly documented and justified. In short, the following suggestions will form the basis for a good proposal:

- Good basic understanding of the fundamental theory.
- Teamwork and co-ordination.
- Good time management.
- Self learning.
- Progressive review with advisor and subject matter experts on proposal progress.

The proposal will demonstrate the team's basic understanding of the concepts and practical implementation of results. Finally with all essential findings documented, verified and proof read, the proposal will be submitted to the competition committee for evaluation and judging. This will provide participants an opportunity to impress the judges.