nzsee
NEW ZEALAND SOCIETY FOR EARTHQUAKE ENGINEERING

# Parallel scalability of OpenSees on NeSI HPC for 2D/3D site response analysis

*M. Eskandarighadi, C.R. McGann, C. Hyde, L. Trow & S. Wang*
Department of Civil and Natural Resources Engineering, University of Canterbury, Christchurch.

## ABSTRACT

This study investigates the parallel scalability of the OpenSees finite element analysis platform in the context of 2D and 3D ground response analyses. A series of 2D and 3D models ranging from 7500 to 480000 elements are developed and analysed using National e-Science Infrastructure (NeSI) high-performance computing (HPC) resources. These models are analysed using two different approaches for running large models in parallel in OpenSees: OpenSeesSP, which performs the parallel domain decomposition as part of the analysis, and OpenSeesMP, which requires the domain decomposition in the input model file. It is demonstrated that the OpenSeesMP tool results in reduced analysis times and increased parallel efficiencies relative to OpenSeesSP for both simple 2D and 3D problems. Trends in the parallel scaling and efficiency are also analysed and recommendations are made for running parallel OpenSees analyses on NeSI HPC resources.

## 1 INTRODUCTION

Throughout history, numerical research could not be done in many fields due to the complex nature of the problems and the lack of powerful computation resources for solving them. However, because of the significant advancements in computational capabilities, it is now possible to study simulations at the desired scale more in-depth and in detail. As the problems get more complicated, the efficiency of calculation becomes a critical factor for massive computation. For addressing this, high performance and scalable calculation tools are required. Improvements in the field of computer science have made parallel computing possible for solving large-scale problems. Parallel computing is a type of calculation where many computations are carried out at the same time, making it an excellent tool for addressing the crucial elements of performance, scalability, and efficiency. Within parallel computing instead of a single CPU with massive processing power, different cores can work simultaneously, where each of them takes part in solving a portion of the problem. It is clear that by dividing the massive solution into small tasks, each task will need less memory than the old fashioned method. With this characteristic, the effective adoption of parallel computation can increase the practicability of large-scale nonlinear site response models.

This study aims to investigate the parallel scalability of OpenSees for both the OpenSeesSP and OpenSeesMP parallel interpreters to find the most efficient combination of cores on NeSI HPC for different

sized models. Furthermore, the goal of this study is to find the more appropriate OpenSees parallel interpreter for use in large-scale analysis.

## 2 PARALLEL ANALYSIS USING OPENSEES

OpenSees is a software framework used for finite element models in earthquake engineering (McKenna et al., 2010; McKenna, 2011), and to cover the parallel computation, it has two interpreters, OpenSeesSP and OpenSeesMP. In OpenSeesSP, the model gets built and analysis will be constructed in one master process which is running the main interpreter and processing the input script. This master process partitions the domain at analysis time, sending information to the remaining parallel cores, so each will take part in solving a portion of the problem. Upon completion of each analysis step, the master process receives and assembles information from the other parallel cores such that the recorders are based on a single cohesive output.

OpenSeesMP was conceived for embarrassingly parallel analysis (e.g. parameter studies), however, it can be used to run large models in parallel. When running a parallel job on OpenSeesMP, each core works as a modified OpenSees interpreter, meaning that each part of model and analysis is conducted separately and there is no master process to handle computations. To use OpenSeesMP in this way, the user needs to partition the domain prior to analysis by providing the model commands to each parallel core by writing in sections in the main input file. This means that OpenSeesMP requires bespoke input model files, e.g. in order to run analyses with both eight cores and sixteen cores it is necessary to have two input files. Each parallel core provides its own output to the recorders so there is also an extra postprocessing step to assemble results. In contrast, OpenSeesSP does not require any modification to the model file to run an analysis on different numbers of parallel cores and all recorded results are returned in a single file.

## 3 CONSIDERED MODELS AND ANALYSIS CASES

A simple one-layered elastic soil model is considered. The base and sides of the model are fixed against out-of-plane translation, while all other nodes are free. The model is analysed for 1000 steps in free-vibration under constant elemental body forces. Model initialisation time comprised < 1% of the execution time for all models. SSPquad and SSPbrick elements (McGann et al., 2012;2015) are used in the 2D and 3D analyses, respectively. Elemental stress and nodal displacements are recorded for all elements and nodes for all analysis steps. Models with 7500, 15000, 30000, 60000, 120000, 240000 and 480000 elements are considered. The 3D models have more degrees of freedom for the same number of elements, so will naturally take longer, but this choice was made to see how things scale in an element-wise manner. The Mumps parallel solver is used and all other analysis parameters are the same for both programs, including use of the linear algorithm with the factorOnce feature and time integration using the TBDF2 scheme. The domain decomposition for the model input in OpenSeesMP interpreter was performed using Scientific ToolKit for OpenSees (STKO) version 1.0.0 (Petracca et al., 2017), which uses Metis for this purpose. Figure 1 shows the 2D and 3D models for the cases with 7500 elements and 16 parallel partitions.

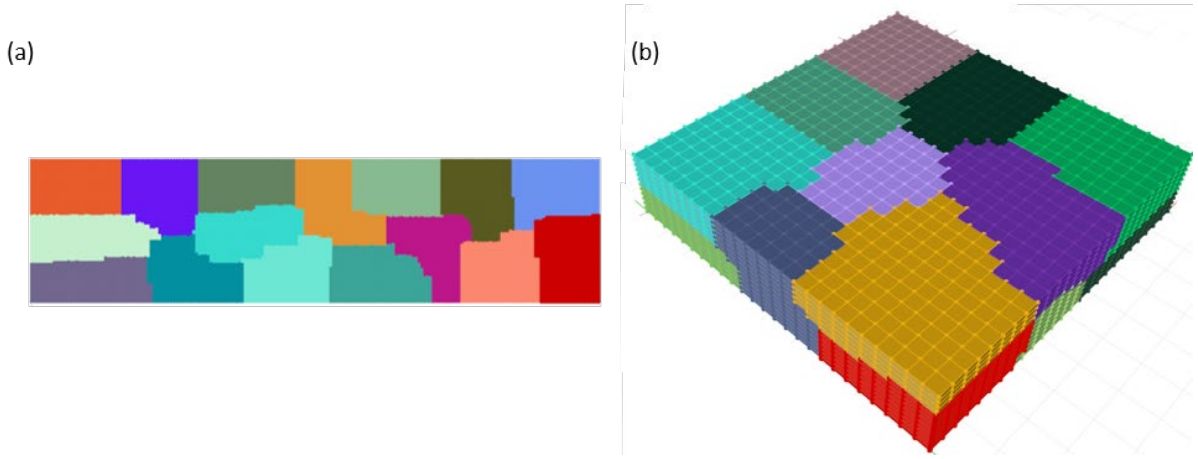


*Figure 1: Schematic of the general model partitioning in (a) 2D analysis, and (b) 3D analysis.*

All analyses were completed on the NeSI Mahuika HPC, which is a Cray CS400 cluster with Intel Xeon Broadwell nodes (NeSI, 2019). The analysis calculations were performed on a series of parallel cores for the purposes of examining the parallel scaling of the OpenSeesSP and OpenSeesMP interpreters. This scaling study was based on analyses made with 1, 2, 4, 8, 16, 32, and 64 cores. Based on memory demands and the particular configuration of the Mahuika compute nodes, it was necessary to run the 32 and 64 core analyses across more than one node. The implications of this are noted in subsequent discussion. Total execution time was recorded for all analyses for the purposes of examining the strong and weak parallel scaling behaviour as is discussed in the following sections.

## 4 STRONG PARALLEL SCALING RESULTS

Strong scaling measures the parallel performance of a program when the size of the problem is held fixed but the number of cores in the parallel analysis is increased, which according to Amdahl (1967) establishes a baseline for parallel performance for fixed-size problems. The decrease in analysis time associated with an increase in cores cannot exceed a factor equal to the number of cores (e.g. using 4 cores cannot be $>$ 4 times faster than a single core). Such linear scaling is rare in practice and is not expected here. Limitations related to parallel communication and the structure of the Mahuika HPC will reduce parallel efficiency, on top of the primary limitations associated with any portion of code that cannot be (or is not) parallelised.

### 4.1 Two-dimensional models

Figure 2 shows the strong parallel speedup performance for the 2D models analysed using both OpenSeesSP and OpenSeesMP. As shown, the speedup in OpenSeesMP is generally superior for all model sizes and number of cores in the parallel analysis, and in most cases it is significantly better. This difference makes sense in the context of Amdahl's law, as the serial portion of an analysis in OpenSeesSP is much larger than for the same analysis in OpenSeesMP and the speedup is correspondingly constrained. Aside from this obvious differences across the two parallel programs, there are some key trends to highlight within the results for each program. The OpenSeesSP speedup curves in Figure 2(a) tend to be flatter, with the speedup for 32 cores generally only slightly greater than that at 16 cores and the performance with 64 cores only marginally worse than at 32 cores. It is also evident that speedup performance is clustered into two general groups, with the relatively smaller models ($\leq$ 60000 elements) displaying similar speedup performance that is notably less than that of the larger models ($\geq$ 120000 elements), which also display similar performance.

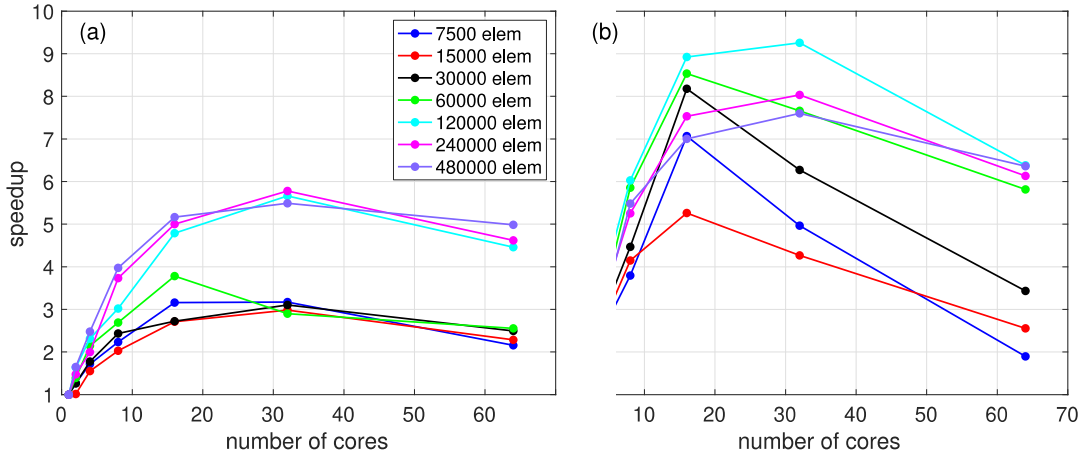


*Figure 2: Strong scaling speedup for 2D analysis models. (a) OpenSeesSP; (b) OpenSeesMP.*

For the OpenSeesMP results of Figure 2(b), there is a tendency for a drop-off in performance for 32 and 64 cores relative to the lesser numbers of cores for most of the models. This effect is almost certainly due to the constraints of the Mahuika system, which is comprised of nodes of CPUs with 36 physical CPU cores per node. Due to this configuration, the 64 core cases had to be run over at least two nodes, and while the 32 core cases would theoretically fit on a single node, it was not possible to run the largest models on a single node for the 32 core cases due to the memory requirements being greater than the available memory. Parallel communication across computational nodes is slower than within a single node, and any speedup associated with an increased number of cores may be offset by the increased overhead associated with this inter-node communication. As shown in Figure 2(b), the smaller model sizes are affected more significantly than the larger model sizes, indicating an expected effect of model size on this trade-off. It is important to note that despite the stronger apparent effect of the parallel communication penalty on the OpenSeeMP models, the speedup in OpenSeesMP is greater than that for OpenSeesSP in all but a single case.

Amdahl's law states that the speedup $S$ in strong scaling can be expressed as

$$S = \frac{1}{1 - p + \frac{p}{n}} \tag{1}$$

where $n$ is the number of cores in the analysis and $p$ is the parallelized portion of the job (with $p + s = 1$ where $s$ is the serial portion of the job). Equation 1 is fit to the strong scaling results shown in Figure 2 in order to estimate the parallel/serial portion for these strong scaling analyses. Weighted nonlinear regression is used for this purpose, with the results for 32 and 64 cores weighted less than the lesser numbers of cores (25% 32/64 cores vs 75% lesser cores) due to the known issues related to the influence of the Mahuika HPC architecture. The parallel portions estimated through this process are shown in Table 1. As shown, these results reinforce the observations made from Figure 2, with OpenSeesMP showing a larger parallel portion across the board and the OpenSeesSP results showing a marked difference for smaller and larger models.

*Table 1: The parallelized portion of analysis from fitting Amdahl's law to 2D model results.*

| Program | 7500 elem | 15000 elem | 30000 elem | 60000 elem | 120000 elem | 240000 elem | 480000 elem |
|---|---|---|---|---|---|---|---|
| OpenSeesSP | 0.68 | 0.62 | 0.66 | 0.74 | 0.82 | 0.84 | 0.85 |
| OpenSeesMP | 0.87 | 0.84 | 0.90 | 0.92 | 0.93 | 0.91 | 0.90 |

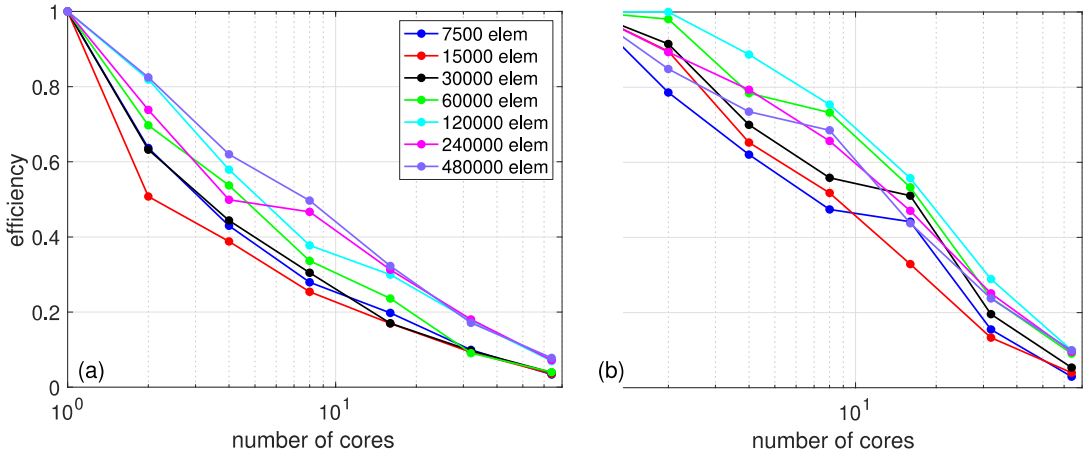


*Figure 3: Strong scaling efficiency for 2D analysis models. (a) OpenSeesSP; (b) OpenSeesMP.*

Figure 3 shows the parallel efficiency of the two programs for the same set of 2D analyses. In this context, parallel efficiency is defined as the ratio of the ideal analysis time to the measured analysis time, where the ideal time for $n$ cores is taken as the measured time for a single core divided by $n$. These parallel efficiency plots show the same information as the speedup plots of Figure 2, but instead of highlighting the reductions in analysis time enabled by using greater numbers of cores, Figure 3 indicates how efficiently the resources are being used to enable this speedup. Perfect efficiency is not expected in either case and while in some sense neither interpreter is particularly efficient (ideally efficiency would be at least 75%), it is clear that the efficiency of OpenSeesMP is superior to OpenSeesSP. There is also a general trend in both interpreters for increasing efficiency with increasing model size, and though this trend does not hold perfectly, it makes sense that the larger models will run more efficiently on greater numbers of cores than smaller models.

## 4.2 Three-dimensional models

The equivalent speedup and efficiency results for the 3D models are shown in Figures 4 and 5, respectively. As shown, the speedup in OpenSeesMP is significantly higher in comparison to OpenSeesSP for all model sizes and number of cores. Figure 4 shows that the speedup in OpenSeesSP tends to improve only slightly in contrast to OpenSeesMP where for larger models (≥30000 elements) the speedup has a significantly higher rate of change up to 32 cores, and for smaller models (≤15000 elements) the speedup has the same quality up to 16 cores. Figure 5 demonstrates the better efficiency of OpenSeesMP to that of OpenSeesSP, however, the general trend of increasing efficiency with increasing model size observed in 2D is not really observed here.

Note that the largest model size considered in 3D is 240000 elements as opposed to the 480000 element maximum size in 2D. The memory demands per CPU for 3D models with 480000 elements run on 1 or 2 cores are in excess of reasonable levels, and it was deemed impractical to run these models just for the purposes of a parallel scaling study. Note that this does not imply that 3D models larger than 240000 elements cannot be run in parallel on the Mahuika HPC, it simply highlights the impracticalities of running such large models in serial analysis. The per-CPU memory demands are more practical for parallel analysis with larger numbers of cores and similar parallel efficiencies to the 240000 element case are expected.

Just as in the 2D cases, Amdahl's law as expressed in Equation 1 is fit to the speedup curves of Figure 4 to assess the respective serial and parallelized portions of the analyses. The resulting parallelized portions for each 3D case are provided in Table 2. As observed with the 2D strong scaling results, the values in Table 2 indicate that the improved speedup of OpenSeesMP is due to a decrease in the serial portion of the analyses.

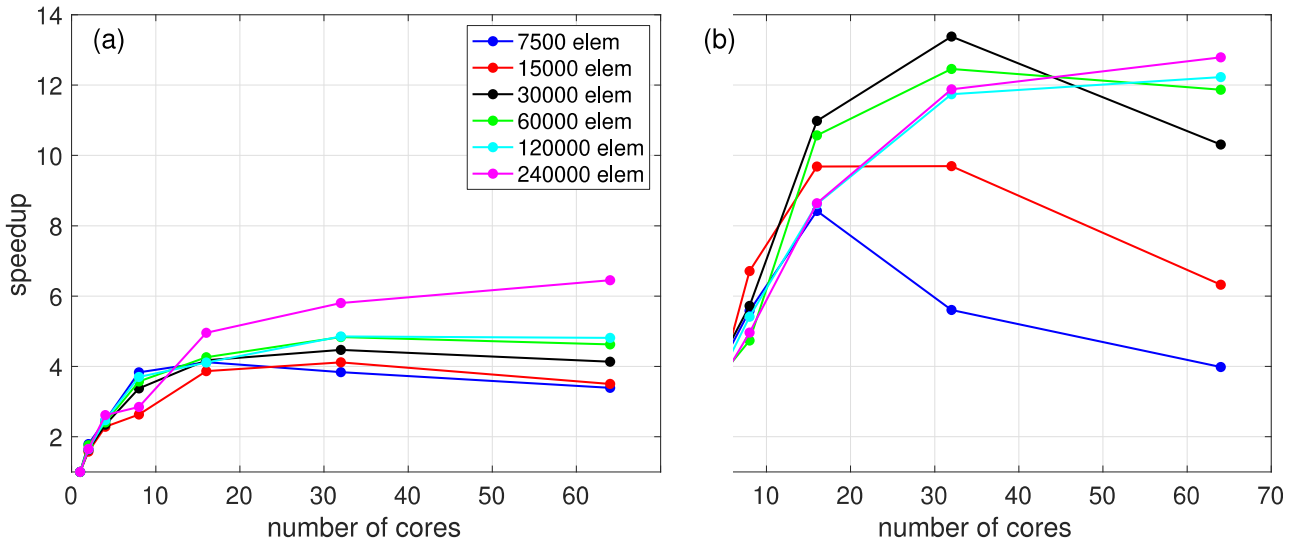


*Figure 4: Strong scaling speedup for 3D analysis models. (a) OpenSeesSP; (b) OpenSeesMP.*

A comparison between 2D and 3D analysis results show that the maximum speedup values in 3D analyses exceed those of the 2D analysis. The element level effort in 3D is greater than for 2D, and it is expected that element-level calculations can be parallelized so speedup in 3D should be greater than in 2D. Tables 1 and 2 also demonstrate that parallelized portions for 3D are generally larger in both OpenSeesSP and OpenSeesMP for all but one case (120000 elements in OpenSeesSP). This confirms the observation that 3D analyses work more efficiently, particularly in OpenSeesMP, where Figure 5 shows that the efficiency values exceed 80% in OpenSeesMP up to 4 cores, while the same values for 2D analysis for 4 cores have an average of 74%.

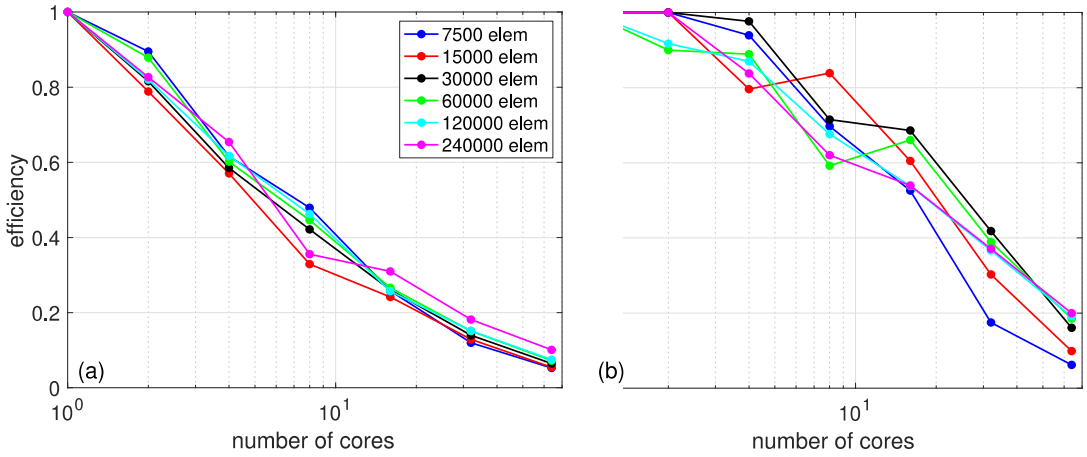


*Figure 5: Strong scaling efficiency for 3D analysis models. (a) OpenSeesSP; (b) OpenSeesMP.*

*Table 2: The parallelized portion of analysis from fitting Amdahl's law to 3D model results*

| Program | 7500 elem | 15000 elem | 30000 elem | 60000 elem | 120000 elem | 240000 elem |
|---|---|---|---|---|---|---|
| OpenSeesSP | 0.77 | 0.76 | 0.79 | 0.81 | 0.81 | 0.85 |
| OpenSeesMP | 0.86 | 0.91 | 0.94 | 0.94 | 0.94 | 0.94 |

## 5 WEAK PARALLEL SCALING RESULTS

Weak scaling measures the performance of a parallel program when the size of the problem is increased in proportion to the increase in the number of cores. In this manner, the portion of work assigned to each core

remains constant as the size of the model is increased. Theoretically perfect weak scaling performance would result in constant analysis time, however, as discussed by Gustafson (1988), the efficiency achieved will be limited by the serial portion of the code that cannot be (or is not) parallelized. Total execution time will generally increase with increasing cores/model size due to these limitations, and based on the known differences in how OpenSeesSP and OpenSeesMP conduct parallel computations, it is expected that OpenSeesMP will be more efficient in weak scaling.

Figures 6 and 7 show the weak scaling results for the 2D and 3D models. As shown, the OpenSeesMP results are both closer to the idealised execution time represented by the dashed line in the left-hand subplots and more efficient than OpenSeesSP. To examine the differences more closely, the results are examined in the context of Gustafson's law, which states that the speedup $S$ for weak scaling can be expressed as

$$S = n + s(1 - n) \tag{2}$$

where $n$ is the number of cores and $s$ is the serial portion of the analysis (non-parallelized portion). By fitting Equation 2 to the weak parallel scaling results using linear regression it is found that the parallelized portion (i.e. $p = 1 - s$) of the analysis is 0.83 for OpenSeesSP in both 2D and 3D, and for OpenSeesMP, $p = 0.86$ and 0.93 for the 2D and 3D analyses, respectively. These results reinforce all of the previous observations made in this study. Parallel analysis in OpenSeesMP reduces the serial portion of the analysis relative to OpenSeesSP (or at least what is interpreted as the serial portion by Gustafson's law), leading to reduced analysis times and more efficient usage of HPC resources. Similar to the strong scaling results, 3D analysis in OpenSeesMP sees a smaller serial portion than 2D analysis with the same number of elements. This indicates that the increased element-level effort in 3D is being parallelized rather than contributing to the serial portion of the job. It is interesting that there is not a similar increase from 2D to 3D for OpenSeesSP. Though this may be attributable to limitations in the curve-fitting, it generally appears from Figures 6 and 7 that the 3D results are only marginally improved, so similar parallel/serial ratios are reasonable.

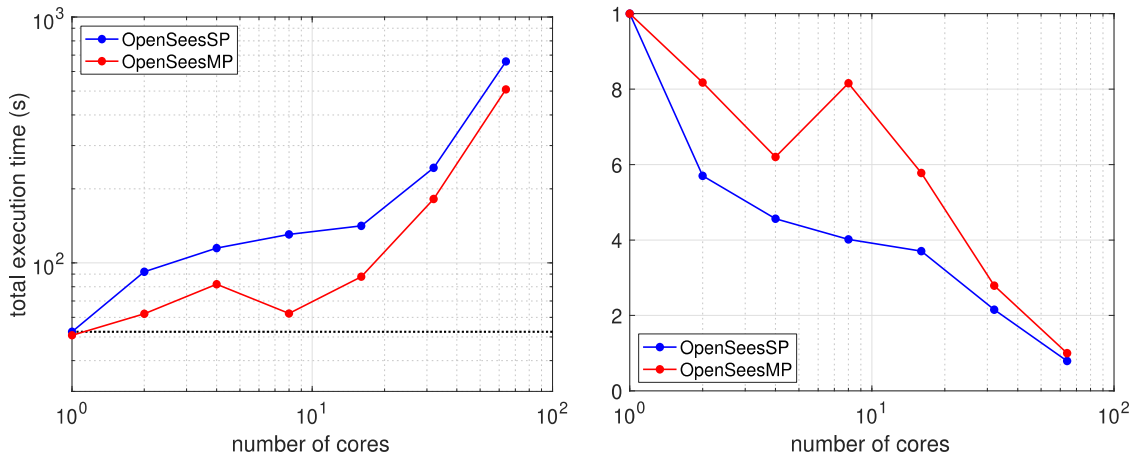


*Figure 6: Weak scaling performance summary for 2D analysis models in OpenSeesSP and OpenSeesMP.*

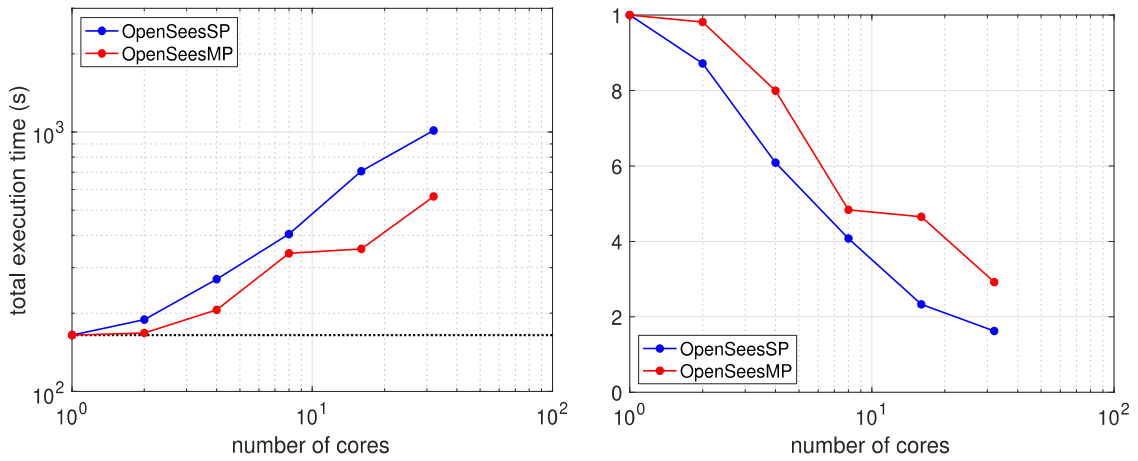


*Figure 7: Weak scaling performance summary for 3D analysis models in OpenSeesSP and OpenSeesMP.*

## 6 CONCLUSIONS

In this paper, the parallel scalability of OpenSees is assessed for 2D and 3D models on NeSI HPC resources through a set of simulations performed on OpenSeesMP and OpenSeesSP. A simple one-layered elastic soil model with gravity analysis is used. The results of this study demonstrate that strong parallel speedup performance in OpenSeesMP is generally better for all model sizes and all number of cores in both 2D and 3D analyses. It was also found that there was a greater difference in performance between the two parallel interpreters in 3D analysis relative to 2D analysis. The OpenSeesSP speedup curves tend to be flatter in most cases and the speedup for 16, 32 and 64 cores is only marginally different in both 2D and 3D analyses. It is also discussed that due to the limitations of the Mahuika system which has 36 physical CPU cores per node, the simulations for 32 and 64 cores could not be performed on one node decreasing the speedup rate of change as the inter-node communication time will add to analysis time. In the 2D analyses, there is a general trend in both interpreters in which efficiency increases as the model size enlarges. This trend is also found in the 3D OpenSeesSP analyses, but not in the 3D OpenSeesMP results.

Weak parallel scaling results based on Gustafson's law illustrate that the serial portion of the OpenSeesMP simulations compared to OpenSeesSP have lesser values, confirming the aforementioned observations. Also, the results show that the increased effort related to the increase in the number of elements in OpenSeesMP is being parallelized rather than devoted to the serial portion of the simulations. It is noteworthy that there is not a strong increase of parallelized portion from 2D to 3D for the same model size in OpenSeesSP. Based on the results of this study, OpenSeesMP is the recommended parallel interpreter for use in very large models when reduced computation time and parallel efficiency are desired.

An important observation that is not shown in these results is the effect of the analysis algorithm on the speedup. The cases considered in this study use the optimal conditions with the Linear algorithm and the factorOnce option. Similar analyses conducted without the factorOnce option or with the Newton algorithm demonstrate more significant parallel scaling, however, the executions times are much longer than under the optimal conditions. Relative to the Newton algorithm the speedup enabled in serial analysis with use of the Linear algorithm is approximately 2 times and with Linear and factorOnce, this speedup is greater than 4. So while the parallel speedup is somewhat suppressed with the optimal options, the execution times are much shorter. The time integration method selected in the analysis was found to impact the execution time, but no consistent trend was observed for any of the implicit integrators in OpenSees, and the differences were modest relative to the parallel speedups observed in the study.

## 7 ACKNOWLEDGEMENTS

The authors wish to acknowledge the use of New Zealand eScience Infrastructure (NeSI) high performance computing facilities and consulting support as part of this research. New Zealand's national facilities are provided by NeSI and funding jointly by NeSI's collaborator institutions through the Ministry of Business, Innovation & Employment's Research Infrastructure programme. This work was supported by the Marsden Fund Council from Government funding, managed by Royal Society Te Apārangi, and QuakeCoRE, a New Zealand Tertiary Commission-funded Centre. This is QuakeCoRE publication number 0647.

## 8 REFERENCES

Amdahl, G.M. 1967. Validity of the single-processor approach to achieving large scale computing capabilities. In *Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, 18-20 April 1967*. New York: Association for Computing Machinery. https://doi.org/10.1145/1465482.1465560

Gustafson, J.L. 1988. Reevaluating Amdahl's Law. *Communications of the ACM*, Vol 31(5) 532-533. https://doi.org/10.1145/42411.42415

McGann, C.R., Arduino, P., & Mackenzie-Helnwein, P. 2012. A stabilized single-point 4-node quadrilateral element for dynamic analysis of fluid saturated porous media. *Acta Geotechnica*, Vol 7(4) 297-311. https://doi.org/10.107/s11440-012-0168-5.

McGann, C.R., Arduino, P., & Mackenzie-Helnwein, P. 2015. A stabilized single-point finite element formulation for three-dimensional dynamic analysis of saturated soils. *Computers and Geotechnics*, Vol 66 126-141. https://doi.org/10.1016/j.compgeo.2015.01.002.

McKenna, F. 2011. OpenSees: A framework for earthquake engineering simulation. *Computing in Science and Engineering*, Vol 13(4) 58-66. https://doi.org/10.1109/MSCE.2011.66

McKenna, F., Scott, M.H., & Fenves, G.L. 2010. Nonlinear finite element analysis software architecture using object composition. *Journal of Computing in Civil Engineering*, Vol 24(1) 95-107. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000002

NeSI. 2019. *Mahuika – NeSI support*. Available at: https://support.nesi.org.nz/hc/en-gb/articles/3600360000163575-Mahuika (Accessed: 1 February 2021)

Petracca, M., Candeloro, F. & Camata, G. 2017. *STKO User Manual*. ASDEA Software Technology, Pescara, Italy.