

Two-level Schwarz Methods using more Subdomains than MPI Processes: Time and Energy Investigations

Martin J. Gander^[0000-0001-8450-9223],
Serge Van Criekingen^[0009-0005-8401-3160]

1 Introduction

The idea of using domain decomposition on computers already emerged when those where only sequential: in 1953 [12]¹, G. Kron, an engineer at General Electric, presented numerical results on a UNIVAC computer using what would now be called a domain decomposition method. In the abstract, the author stated the following:

A set of principles and a systematic procedure are presented to establish the exact solutions of very large and complicated physical systems, without solving a large number of simultaneous equations and without finding the inverse of large matrices. The procedure consists of tearing the system apart into several smaller component systems. After establishing and solving the equations of the component systems, the component solutions themselves are interconnected to obtain outright, by a set of transformations, the exact solution of the original system.

Nowadays, supercomputing domain decomposition experiments typically use one MPI process per CPU processor core (and we use this setting also in this work), and as many CPU cores as subdomains, each of the CPU core handling one subdomain. However, using more subdomains than CPU cores, i.e., having each core handle more than one subdomain, can be beneficial, as was already shown by G. Kron for one processor core, and as can nowadays be shown on several thousands of processors on modern parallel computers. This was done in a nuclear engineering context in [14], where the interest of domain decomposition as a sequential method is shown, as well

Martin J. Gander
University of Geneva, Switzerland, email: martin.gander@unige.ch

Serge Van Criekingen
Institut du Développement et des Ressources en Informatique Scientifique (IDRIS), CNRS, Université Paris-Saclay, F-911403, Orsay, France & Université Paris-Saclay, UVSQ, CNRS, CEA, Maison de la Simulation, 91191, Gif-sur-Yvette, France, email: serge.van.criekingen@idris.fr

¹ We thank David E. Keyes for pointing this reference out to us.

as the existence of an optimal number of subdomains to use for a given number of CPU cores.

In this work, we use the possibility given by PETSc [3, 4, 5] to use more subdomains than CPU cores for the Restricted Additive Schwarz (RAS, [6]) method: PETSc's (one-level) RAS implementation, named PCASM, offers in this view the `-pc_asm_blocks` option. Our model problem will be the Laplace problem in 2D and 3D (respectively discretized with 5- and 7-point finite differences), that we solve with two-level RAS preconditioning and GMRES acceleration (with restart parameter fixed to 30), starting from a random initial guess and converging to the zero solution with a relative convergence tolerance of $1.e-8$. Besides RAS, its optimized version ORAS [10], which uses optimized interface conditions, will also be considered. In all our experiments, the overlap is fixed to 1 in the PETSc sense, i.e., an algebraic overlap of 2. For details on our implementation of two-level RAS/ORAS in PETSc, see [7].

The coarse spaces considered in this work are the Nicolaidis coarse space [13], which uses a constant basis function on each subdomain, and the Merged.Q1 coarse space (Merged_1 of [8]), which uses a single coarse function at each cross point of the non-overlapping decomposition (obtained by merging the Q1 corresponding coarse functions) and is meant to be used with GMRES acceleration or with relaxation, avoiding duplication of effects in resolving the RAS iteration operator eigenmodes.

Besides time-to-solution, our numerical results also include energy-to-solution measurements, a subject of increasing importance in supercomputing centers. Note that these measurements are obtained at the computing node level, so that they do not take into account other aspects such as internal cooling, storage, network switches and Power Usage Efficiency (PUE) of the datacenter.

In Section 2 we present 2D strong scaling results for the two-level RAS method, that is, we use a fixed problem size and progressively increase the number of CPU cores used. This is done for a varying number of subdomains, using one or more subdomains per core, i.e., per MPI process. In Section 3 we present strong scaling 3D results for the two-level ORAS method, and moreover compare them to algebraic multigrid results.

Our results were obtained on the CPU ("Genoa") partition of the HPE-Cray Adas-tra machine hosted at CINES in Southern France (www.cines.fr). Each compute node hosts two AMD Genoa EPYC 9654 processors each with 96 processor cores (so a total of 192 cores per node) at 2.4 GHz, with 768 GB memory per node. The energy measurements were obtained by accessing the energy counter of each of the involved nodes at the beginning and at the end of each run.

2 2D RAS results

We consider our model Laplace problem with size fixed to 2^{26} , i.e, a 8192×8192 fine mesh, using N_{CPU} processor cores with $N_{\text{CPU}} = 64, 256, 1024$ and 4096 . For each N_{CPU} , we use N_{Sub} subdomains with $N_{\text{Sub}} = N_{\text{CPU}}, 4 N_{\text{CPU}}, 16 N_{\text{CPU}}, 64 N_{\text{CPU}}, \dots$ up

Coarse space type	N_{Sub}	coarse space size	iterations
Nicolaidis	64	64	249
	256	256	195
	1024	1024	131
	4096	4096	90
	16,384	16,384	66
	65,536	65,536	47
Merged_Q1	64	49	109
	256	225	78
	1024	961	57
	4096	3969	42
	16,384	16,129	30
	65,536	65,025	23
	262,144	261,121	17

Table 1: Number of iterations and coarse space sizes for each number of subdomains N_{Sub} .

to 262,144 (2^{18}). Given the problem size fixed to 2^{26} , this corresponds to using local meshes of size 1024×1024 (with 64 subdomains) to 16×16 (with 2^{18} subdomains).

With our coarse space choice, increasing the number of subdomains increases the coarse space size, so we expect this to lower the number of GMRES iterations to be performed. Table 1, which gives the number of iterations as a function of the number of subdomains and coarse space size, shows that this is true for our test problem. Note also the striking superiority of the Merged_Q1 coarse space over the Nicolaidis one in this case: about half the number of iterations with a similar (even slightly smaller) coarse space size.

While using more than one subdomain per CPU core implies thus less iterations, it also implies that several small local solves (LU) are to be performed sequentially on each core, instead of a single (larger) one. However, the combined total time of several small solves is typically smaller than a large one, since their complexity decreases more than linearly with the problem size². On the other hand, increasing the coarse space size means a larger coarse problem to solve and more local submatrices to create. With opposite effects, we can expect that there is an optimal number of subdomains N_{Sub} to be found for any fixed N_{CPU} .

Fig. 1 shows the different contributions to the total computing time using the Merged_Q1 coarse space, mainly distinguishing between RAS and coarse correction (cc) contributions, each split into setup and solution phase. As expected, we observe that the RAS setup time increases with the number of subdomains for a given number of CPU cores since there are more and more local submatrices to create. In fact, the RAS setup time accounts for the total computing time at 262,144 subdomains for $N_{\text{CPU}} = 64$, and it is nearly the case for $N_{\text{CPU}} = 256$. The coarse correction setup time

² The same idea was already present in [12]: *If a physical system is divided into n parts, the inversion of each part consumes $1/n^3$ of the inversion time of the original system. Hence the n inversions consume $1/n^2$ of the original time.*

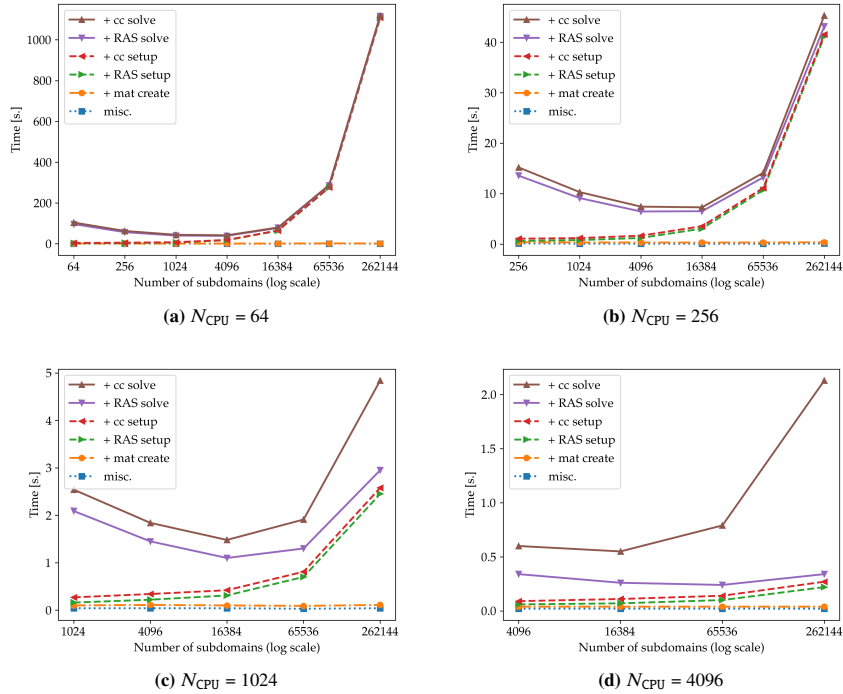


Fig. 1: Detailed computing times for a fixed number of CPU cores and varying number of subdomains (cc = coarse correction) using the Merged.Q1 coarse space.

appears insignificant in all cases. As for the solving times, we see that the RAS solving time decreases when increasing the number of subdomains, because there are less iterations to perform (better coarse space) and also, as announced, because several sequential small local solves are faster than a large local solve: for $N_{\text{CPU}} = 1024$, at $N_{\text{Sub}} = 1024$, the 57 iterations are performed in 1.82s, which means .032s per iteration, while at $N_{\text{Sub}} = 262,144$, the 17 iterations are performed in .37s, which means .022s per iteration. The time per iteration thus decreases when several smaller local solves are performed. Finally the coarse correction solving time logically increases with the size of the coarse space and thus with the number of subdomains. Summing up all the contributions, we have that for each fixed number of CPU cores, there is an optimal number of subdomains to minimize the total computing time (i.e., time-to-solution), as expected.

Fig. 2(a) shows the total time-to-solution for the different values of N_{CPU} on a same plot, including also the results using the Nicolai des coarse space. From the zoomed version (Fig. 2(c)), we conclude that optimal time-to-solution is obtained using 4096 CPU cores and 16384 subdomains. Fig. 2(b) and 2(d) show the corresponding energy-to-solution results. The optimal energy-to-solution is obtained using 1024 CPU cores and 16384 subdomains.

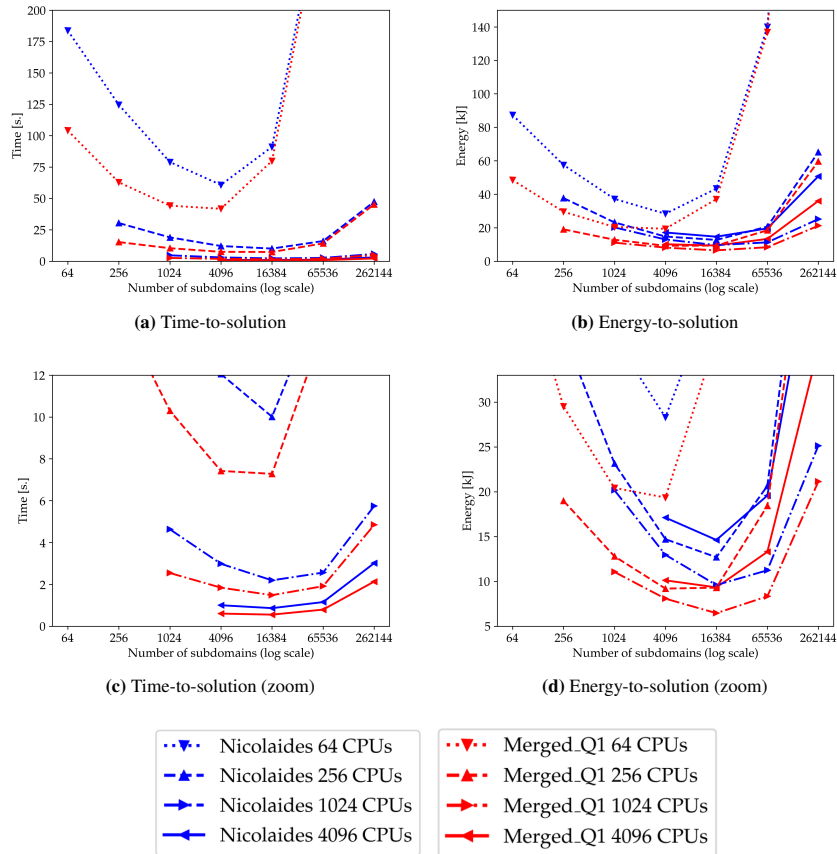


Fig. 2: Total RAS time- and energy-to-solution for fixed N_{CPU} and varying number of subdomains using the Nicolaides (in blue) and Merged_Q1 (in red) coarse spaces. Note that “CPU” stands for “CPU cores” in the legend.

Besides time- and energy-to-solution, we also provide later in Fig 4(a) the total memory used by the jobs per MPI process, i.e., per CPU core. (Note that PETSc’s `-mat_increase_overlap_scalable` option is used for all the runs). Using more cores for the same problem size logically reduces the amount of memory used by each of them (- between 256 and 64 cores we roughly observe the same factor of 4 for the memory, and for more cores the incompressible memory use per core predominates).

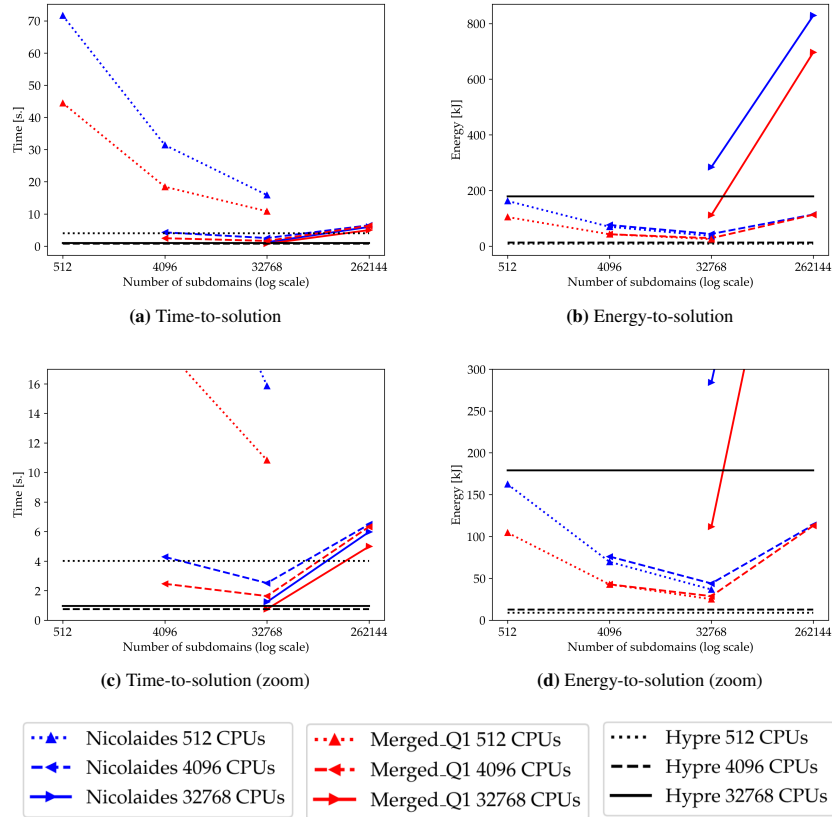


Fig. 3: Total ORAS time- and energy-to-solution in 3D for fixed N_{CPU} and varying number of subdomains using the Nicolaides (in blue) and Merged.Q1 (in red) coarse spaces, together with equivalent HYPRE/BoomerAMG results (in black). Note that “CPUs” stands for “CPU cores” in the legend.

3 3D ORAS results and comparison with multigrid

We now consider our model problem in 3D with a problem size fixed to 2^{27} , i.e., a $512 \times 512 \times 512$ fine mesh. This time we use optimized RAS [10], again with the Nicolaides and Merged.Q1 coarse spaces, and compare our results to the HYPRE/BoomerAMG algebraic multigrid preconditioning [11] option available through PETSc (with optimized tuning from [15]). We use $N_{\text{CPU}} = 512, 4096$ and $32,768$ and, for each N_{CPU} , $N_{\text{Sub}} = N_{\text{CPU}}, 8 N_{\text{CPU}}, 64 N_{\text{CPU}}, \dots$ up to $262,144$ (2^{18}). Given the problem size fixed to 2^{27} , this corresponds to using local meshes of size $128 \times 128 \times 128$ (with 64 subdomains) to $8 \times 8 \times 8$ (with 2^{18} subdomains).

Total time- and energy-to-solution are shown in Fig. 3. The curves at 512 CPU cores are stopped at 32,768 subdomains because of an up-to-now unexplained

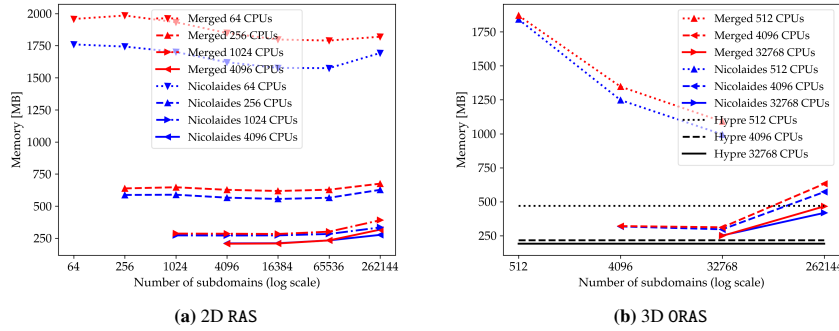


Fig. 4: Physical memory used per MPI process. Note that “CPUs” stands for “CPU cores” in the legend.

error occurring in MUMPS [1, 2] (used as local direct solver by PETSc) during the local subdomain LU factorizations. We observe here that, for both Nicolaides and Merged_Q1 coarse spaces, the optimal time-to-solution is obtained using 32,768 cores and as many subdomains, while the optimal energy-to-solution is obtained using 512 cores and 16,384 subdomains. As for HYPRE/BoomerAMG, the optimal time-to-solution is obtained at 4096 cores and is comparable to our best ORAS results, while the optimal energy-to-solution is obtained at 512 cores and remains better than our best ORAS results (8.704 vs. 25.041 kJ). Note that the performances of our ORAS method could potentially be further improved by using a multilevel (rather than just two-level) approach as developed in [9].

Finally, Fig. 4(b) shows the equivalent memory plots. Unlike what happens in 2D, we observe in 3D that an optimal N_{Sub} , N_{CPU} combination also exists for the memory consumption and that it tends to first decrease when increasing the number of subdomains. Actually, trying to solve our 2^{27} problem with only 64 CPU cores (not shown on the plot), we observed that using 64 and 512 subdomains then yields an out-of-memory error, but not using 4096. On a given machine, adjusting the number of subdomains can thus make a problem solvable by lowering the required memory.

4 Conclusion

Using more subdomains than CPU cores can be useful, with an optimal to be found in terms of time, energy and memory. The optimal combination of number of subdomains and number of cores can be very different according to the criteria chosen. While the memory limitation is fixed by the hardware, the choice between optimizing in time or in energy is more of a strategic nature. It should take into account the global energy consumption at the infrastructure level and not only at the node level, as well

as the ratio between the operational and hardware manufacturing/decommissioning energetic costs and related emissions.

Acknowledgements This work was performed using HPC resources from GENCI-CINES.

References

1. Amestoy, P.R., Buttari, A., L'Excellent, J.Y., Mary, T.: Performance and scalability of the block low-rank multifrontal factorization on multicore architectures. *ACM Transactions on Mathematical Software* **45**(1), 2:1–2:26 (2019)
2. Amestoy, P.R., Duff, I.S., Koster, J., L'Excellent, J.Y.: A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications* **23**(1), 15–41 (2001)
3. Balay, S., Abhyankar, S., Adams, M.F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E.M., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W.D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M.G., Kong, F., Kruger, S., May, D.A., McInnes, L.C., Mills, R.T., Mitchell, L., Munson, T., Roman, J.E., Rupp, K., Sanan, P., Sarich, J., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., Zhang, J.: PETSc Web page. <https://petsc.org/> (2025). URL <https://petsc.org/>
4. Balay, S., Abhyankar, S., Adams, M.F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E.M., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W.D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M.G., Kong, F., Kruger, S., May, D.A., McInnes, L.C., Mills, R.T., Mitchell, L., Munson, T., Roman, J.E., Rupp, K., Sanan, P., Sarich, J., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., Zhang, J.: PETSc/TAO users manual. Tech. Rep. ANL-21/39 - Revision 3.23, Argonne National Laboratory (2025)
5. Balay, S., Gropp, W., McInnes, L.C., Smith, B.: Efficient management of parallelism in object oriented numerical software libraries. In: E. Arge, A.M. Bruaset, H.P. Langtangen (eds.) *Modern Software Tools in Scientific Computing*, pp. 163–202. Birkhäuser Press (1997)
6. Cai, X.C., Sarkis, M.: A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comp.* **21**(2), 239–247 (1999)
7. Gander, M., Van Criekingen, S.: New coarse corrections for restricted additive Schwarz using PETSc. In: *Domain Decomposition Methods in Science and Engineering XXV, Lecture Notes in Computational Science and Engineering*, pp. 483–490. Springer-Verlag (2019)
8. Gander, M., Van Criekingen, S.: Merged Q1 coarse spaces for Schwarz methods in 2D and 3D. In: *Domain Decomposition Methods in Science and Engineering XXVIII, Lecture Notes in Computational Science and Engineering*. Springer-Verlag (In press)
9. Gander, M., Vanzan, T.: Multilevel optimized Schwarz methods. *SIAM J. Sci. Comput.* **42**(5), A3180–A3209 (2020)
10. Gander, M.J.: Optimized Schwarz methods. *SIAM J. Numer. Anal.* **44**(2), 669–731 (2006)
11. Henson, V., Yang, U.: Boomeramg: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics* **41**, 155–177 (2002)
12. Kron, G.: A set of principles to interconnect the solutions of physical systems. *Journal of Applied Physics* **24**(8), 965–980 (1953)
13. Nicolaidis, R.A.: Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis* **24**, 355–365 (1987)
14. Van Criekingen, S., Nataf, F., Havé, P.: Parafish: a Parallel $FE - P_N$ Neutron Transport Solver based on Domain Decomposition. *Annals of nuclear energy* **38**(1), 145–150 (2011)
15. Vassilevski, P.S., Yang, U.M.: Reducing communication in algebraic multigrid using additive variants. *Numer. Linear Algebra Appl.* **21** (2), 275–296 (2014)