Technical Report Series
Center for Data and Simulation Science

Axel Klawonn, Martin Lanser, Oliver Rheinbach, Gerhard Wellein, Markus Wittmann

Energy Efficiency of Nonlinear Domain Decomposition Methods

Technical Report ID: CDS-2018-3
Available at `http://kups.ub.uni-koeln.de/id/eprint/8654`
Submitted on October 2, 2018

ENERGY EFFICIENCY OF NONLINEAR DOMAIN DECOMPOSITION METHODS*

A. KLAWONN[†‡], M. LANSER[†‡], O. RHEINBACH[§], G. WELLEIN[¶], AND M. WITTMANN[‖]

Abstract. A nonlinear domain decomposition (DD) solver is considered with respect to improved energy efficiency. In this method, nonlinear problems are solved using Newton's method on the subdomains in parallel and in asynchronous iterations. The method is compared to the more standard Newton-Krylov approach, where a linear domain decomposition solver is applied to the overall nonlinear problem after linearization using Newton's method. It is found that in the nonlinear domain decomposition method, making use of the asynchronicity, some processor cores can be set to sleep to save energy and to allow better use of the power and thermal budget. Energy savings up to 77% are observed compared to the more traditional Newton-Krylov approach, which is synchronous by design, using up to 5120 Intel Broadwell (Xeon E5-2630v4) cores. The total time to solution is not affected. On the contrary, remaining cores of the same processor may be able to go to turbo mode, thus reducing the total time to solution slightly. Last, we consider the same strategy for the ASPIN (Additive Schwarz Preconditioned Inexact Newton) nonlinear domain decomposition method and observe a similar potential to save energy.

Key words. Nonlinear FETI-DP, Nonlinear Domain Decomposition, Nonlinear Elimination, Newton's method, Nonlinear Problems, Parallel Computing, Energy Efficiency, LIKWID

AMS subject classifications. 68W10, 68U20, 65N55, 65F08, 65Y05

1. Introduction.

In recent years, many nonlinear domain decomposition approaches have been introduced and their superiority over the classical combination of a nonlinear solver, e.g., Newton's method, with a linear domain decomposition approach has been shown for many model problems [4–6,8,11,12,21,22,24–26,34–36,38]. Nonlinear domain decomposition methods are solution approaches for nonlinear problems and apply the concepts and ideas of linear domain decomposition methods as, e.g., Overlapping Schwarz [4–6, 8, 11, 12, 34–36], FETI (Finite Element Tearing and Interconnecting) [38], FETI-DP (Finite Element Tearing and Interconnecting - Dual-Primal) [22, 24, 26], and BDDC (Balancing Domain Decomposition by Constraints) [21, 22, 26] directly to a nonlinear problem before a nonlinear solver is applied. Although all these methods behave quite differently in their linear as well as nonlinear variants, they have some common features and properties. We will show that one of these properties makes nonlinear domain decomposition methods typically more energy efficient compared with their linear relatives embedded in some nonlinear solver. This effect is not only caused by a shorter runtime but also the power

*This work was supported in part by Deutsche Forschungsgemeinschaft (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) under grants KL 2094/4-1, KL 2094/4-2, RH 122/2-1, and RH 122/3-2.

[†]Department of Mathematics and Computer Science, University of Cologne, Weyertal 86-90, 50931 Köln, Germany, axel.klawonn@uni-koeln.de, martin.lanser@uni-koeln.de, url: http://www.numerik.uni-koeln.de

[‡]Center for Data and Simulation Science, University of Cologne, Germany, url: http://www.cds.uni-koeln.de

[§]Institut für Numerische Mathematik und Optimierung, Fakultät für Mathematik und Informatik, Technische Universität Bergakademie Freiberg, Akademiestr. 6, 09596 Freiberg, oliver.rheinbach@math.tu-freiberg.de, url: http://www.mathe.tu-freiberg.de/nmo/mitarbeiter/oliver-rheinbach

[¶]Department of Computer Science, Friedrich-Alexander University Erlangen-Nürnberg, Martensstr. 1, 91058 Erlangen, Germany gerhard.wellein@fau.de, url: https://hpc.fau.de

[‖]Erlangen Regional Computing Center, Friedrich-Alexander University Erlangen-Nürnberg, Martensstr. 1, 91058 Erlangen, Germany markus.wittmann@fau.de

consumption is typically lower during most of the computation. In this paper, we will show and explain this effect in detail for the example of nonlinear FETI-DP methods. First we will provide a more general description of the concept and show that it is also applicable to other methods as, e.g., ASPIN (Additive Schwarz Preconditioned Inexact Newton) [4].

Given is a discrete nonlinear problem

$$A(x) = 0, \tag{1}$$

which has been obtained by a discretization of a nonlinear partial differential equation. In contrast to classical approaches, e.g., Newton-Krylov-DD (Domain Decomposition) methods, where (1) is linearized by Newton's method and the tangential system is solved iteratively using a domain decomposition approach, the discrete nonlinear function (1) is replaced by an alternative formulation

$$\mathcal{A}(x) = 0. \tag{2}$$

Here, $\mathcal{A}$ is obtained by applying domain decomposition strategies, i.e., by decomposing (1) into local nonlinear problems on subdomains. These local nonlinear problems are decoupled or, for numerical scalability, weakly coupled.

Of course, it has to be guaranteed that (1) and (2) have the same solution. Then, instead of (1), equation (2) is solved by a nonlinear solver. If $\mathcal{A}$ is designed properly, the nonlinear solver takes fewer steps to solve the problem and therefore the time to solution is reduced. Another advantage often observed is increased robustness [4, 21, 35]. In general, current methods to form $\mathcal{A}$ can be interpreted as a nonlinear right-preconditioning approach

$$\mathcal{A}(x) := A(M(x)) \tag{3}$$

or as a a nonlinear left-preconditioning approach

$$\mathcal{A}(x) := G(A(x)). \tag{4}$$

Examples for the latter case are ASPIN [4] and RASPEN [8], while Nonlinear-FETI-DP and Nonlinear-BDDC can be interpreted as nonlinearly right preconditioned methods; see [26]. To be efficient, applications of the preconditioners M and G should be cheap compared with an application of A. Both should put the initial value near to the solution and to obtain the correct solution, one has to ensure that $G(0) = 0$. Let us remark that in the case of nonlinear right preconditioning, we search the solution $M(x)$ instead of x. In most methods, the functions M or G are not given explicitly, but only implicitly by defining local nonlinear problems on subdomains. These local problems are decoupled or, sometimes, coupled through a small coarse space. When solving (2) with Newton's method, the function G or M has to be evaluated in each Newton step, and therefore all the local nonlinear problems have to be solved, e.g., by local Newton iterations. However, these local iterations are easily parallelizable and can often be performed desynchronized among the processors. A rough sketch of a nonlinear domain decomposition method is given in Figure 1.

However, some of the local nonlinear problems in G or M can converge fast or even in a single step (nearly linear behavior), while others may take many Newton steps to converge. Under the assumption that each computational core solves exactly one local nonlinear problem, problem dependent load imbalances can thus arise. This can be tolerated as long as the time to solution is faster than using classical approaches

Start with some initial value
Iterate until convergence
    /* Evaluation of G or M */
    Solve local nonlinear problems with Newton's method (inner Newton loop)
    Solve linearized problem
    Update solution
End of outer Newton loop

FIG. 1. *Sketch of a nonlinear domain decomposition method using Newton's method.*

with proper load balance, but is nonetheless a topic of current research. Removing the load imbalance completely without loosing convergence properties is difficult since a redistribution of the work, e.g., by resizing subdomains changes the nonlinear problem $\mathcal{A}$ completely. However, computational approaches such as the superman strategy [16] can help.

In this paper, we investigate a different approach and decide to set cores to sleep, when they have finished solving their local nonlinear problem until all other cores catch up. This can save energy compared to classical approaches and is feasible because the desynchronization of the local Newton iterations in nonlinear DD methods results in potential sleep times at the order of seconds; see subsection 4.2. To prove this effect, we compare the classical Newton-Krylov-FETI-DP approach with Nonlinear-FETI-DP-3 with respect to scalability, runtime, load balancing, energy consumption, and power efficiency. Let us remark here that Nonlinear-FETI-DP-3 is a good testing prototype, since completely decoupled local nonlinear problems are solved in each outer Newton step. Nonetheless, all introduced concepts can be carried over to different nonlinear DD methods and, even more generally, to many other applications with a severe load imbalance.

The remainder of the paper is organized as follows. In section 2, in order to provide a self-contained paper, we give a detailed description of Newton-Krylov-FETI-DP and Nonlinear-FETI-DP-3 with a focus on the local solves and the load imbalance in the latter one. We describe the investigated model problems in section 3 and our approach to save energy as well as corresponding measurements in section 4. In section 5 we provide a model to interprete the measurements results from section 4.

2. An Introduction to Nonlinear FETI-DP Methods. In this section, we briefly introduce the nonlinear FETI-DP (Finite Element Tearing and Interconnecting - Dual Primal) framework (see [23,26] for a detailed description) and derive Nonlinear-FETI-DP-3, the variant we use in this paper to represent nonlinear domain decomposition methods. As already mentioned above, instead of solving (1) directly by applying Newton's method, in nonlinear FETI-DP methods (1) is replaced by the equivalent formulation

$$\mathcal{A}(x) := A_{NL}\left(M(\tilde{u}, \lambda)\right) = 0, \tag{5}$$

which is then solved by a Newton-Krylov approach. Here, A_{NL} is a nonlinear saddle point formulation and M a nonlinear preconditioner, both derived by transferring the ideas of the linear FETI-DP domain decomposition approach [9, 10, 27–30] to the nonlinear problem (1). If constructed properly and M provides certain properties (see [26]), this approach can reduce the time to solution. Additionally, the application of M, which has to be performed in each Newton step, contains a lot of local work

and is easily parallelizable. On the other hand, an application of M can introduce a strong load imbalance; see subsections 2.3 and 2.4 for details.

Let us first summarize linear FETI-DP and introduce the classical Newton-Krylov-FETI-DP method in subsection 2.1 before we proceed to describe the nonlinear variant in subsection 2.3.

2.1. Newton-Krylov-FETI-DP. In general, we assume that (1) is obtained by a finite element discretization of a partial differential equation defined on a computational domain $\Omega \subset \mathbb{R}^d,\ d = 2, 3$. We denote our finite element space, which discretizes Ω, by V^h and can obviously rewrite (1) as

$$\widehat{K}(\hat{u}) = \hat{f} \Leftrightarrow A(\hat{u}) := \widehat{K}(\hat{u}) - \hat{f} = 0, \tag{6}$$

where $\widehat{K} : V^h \to V^h$ is nonlinear in $\hat{u} \in V^h V^h$ and $\hat{f} \in V^h$ is independent of $\hat{u}$; see also section 3 for an example considering a model problem. We replaced x from (1) by $\hat{u}$ to match with the notation of nonlinear FETI-DP in the literature. In Newton-Krylov-FETI-DP, (6) is solved using Newton's method, i.e., by the iteration $\hat{u}^{(k+1)} = \hat{u}^{(k)} - \delta\hat{u}^{(k)}$ with some initial value $\hat{u}^{(0)}$ and the updates defined by

$$D\widehat{K}(\hat{u}^{(k)})\,\delta\hat{u}^{(k)} = \widehat{K}(\hat{u}^{(k)}) - \hat{f}. \tag{7}$$

Here, $D\widehat{K}(\hat{u}^{(k)})$ is the Jacobian matrix of $\widehat{K}$ evaluated in $\hat{u}^{(k)}$. Finally, in Newton-Krylov-FETI-DP, a linear FETI-DP domain decomposition approach is chosen to solve the linearized system (7), which occurs in each Newton step, iteratively. This automatically classifies Newton-Krylov-FETI-DP as an inexact Newton method.

Let us briefly describe linear FETI-DP and introduce some necessary notation. We assume to have a decomposition of Ω into $N \in \mathbb{N}$ nonoverlapping subdomains $\Omega_i,\ i = 1, ..., N$, i.e., $\Omega = \bigcup_{i=1}^{N} \Omega_i$. Each Ω_i is the union of finite elements; see also Figure 2 (left). The finite element subspaces associated with $\Omega_i,\ i = 1, ..., N$, are denoted by $W_i,\ i = 1, ..., N$. We obtain local nonlinear finite element problems $K_i(u_i) - f_i = 0$ with $K_i : W_i \to W_i$ and $f_i \in W_i$ by restricting the considered differential equation to Ω_i and discretizing its variational formulation using the finite element spaces W_i; see also section 3 for an example using a model problem. The local Jacobian matrices belonging to $K_i(\cdot)$ are denoted by $DK_i(\cdot)$. With restrictions $R_i : V^h \to W_i,\ i = 1, ..., N$, $R^T := \left(R_1^T, ..., R_N^T\right)$, $u^T := \left(u_1^T, ..., u_N^T\right)$, $K(u)^T := \left(K_1(u_1)^T, ..., K_N(u_N)^T\right)$, and $DK(u) = \operatorname{diag}\left(DK_1(u_1), ..., DK_N(u_N)\right)$, we have the identities

$$\widehat{K}(\hat{u}) = R^T K(R\hat{u}) \tag{8}$$

and

$$D\widehat{K}(\hat{u}) = R^T DK(R\hat{u})R. \tag{9}$$

The application of R^T in (8) and (9) has thus the effect of a finite element assembly of local finite element functions on the interface $\Gamma := \left(\bigcup_{i=1}^{N} \partial\Omega_i\right) \setminus \partial\Omega$; see Figure 2.

Let us assume, we have sorted and decomposed a solution u from the decoupled space $W = W_1 \times ... \times W_n$ into interface variables u_Γ and all remaining interior variables u_I, i.e., $u^T = \left(u_I^T, u_\Gamma^T\right)$. In FETI-DP, we further subdivide the degrees of freedoms on the interface u_Γ into primal variables u_Π and dual variables u_Δ. Let us remark that all these variables are still local to the subdomains, e.g., we have $u_\Pi^T = (u_{1_\Pi}^T, ..., u_{N_\Pi}^T)$.

Here, $u_{i_\Pi}, i = 1, ..., N$ is the vector of primal solutions on subdomain $\Omega_i, i, ..., N$. We now introduce another assembly operator R_Π^T, similar to R^T, which assembles only in the primal variables. We denote the corresponding primally assembled finite element space with $\widetilde{W}$; see Figure 3 (right) for an illustration of $\widetilde{W}$, where subdomain vertices are chosen to be primal. Therefore, we have $R_\Pi : \widetilde{W} \to W$ and any $\tilde{u} \in \widetilde{W}$ has the structure $\tilde{u}^T = \left(u_I^T, u_\Delta^T, \tilde{u}_\Pi^T\right)$, where $\tilde{u}_\Pi$ is now a vector of global variables and will constitute our global coarse problem or second level problem. We define the primally coupled operators by

$$\widetilde{K}(\tilde{u}) = R_\Pi^T K(R_\Pi \tilde{u}) \tag{10}$$

and

$$D\widetilde{K}(\tilde{u}) = R_\Pi^T DK(R_\Pi \tilde{u}) R_\Pi. \tag{11}$$

Enforcing continuity in the dual variables is done by enforcing $B\tilde{u} = 0$, using a linear jump operator B (see [27] for a detailed definition of B) and Lagrange multipliers. We obtain the equation system

$$\begin{pmatrix} D\widetilde{K}(\tilde{u}^{(k)}) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta\tilde{u}^{(k)} \\ \lambda \end{pmatrix} = \begin{pmatrix} \widetilde{K}(\tilde{u}^{(k)}) - \tilde{f} \\ 0 \end{pmatrix} \tag{12}$$

which is equivalent to (7) and where λ is the vector of the Lagrange multipliers. Of coarse, several dual variables always belong to a common physical node on the interface and deliver more than a single entry in $\tilde{u}$. Nevertheless, after solving (12), continuity is guaranteed on the interface since $B\tilde{u} = 0$ is enforced and all the dual variables belonging to the same physical node hold the same value. Therefore, the solution $\delta\hat{u}^{(k)}$ of (7) can be easily obtained from $\delta\tilde{u}^{(k)}$ using the restriction $R_D : V^h \to W$, which is a weighted variant of R and the weights are defined as the inverses of the multiplicities of the dual variables. We then have $\delta\hat{u}^{(k)} = R_D^T R_\Pi \delta\tilde{u}^{(k)}$.

By a block elimination in (12) we derive the system

$$F(\tilde{u}^{(k)})\lambda = d(\tilde{u}^{(k)}) \tag{13}$$

with $F(\tilde{u}^{(k)}) = -B\left(D\widetilde{K}(\tilde{u}^{(k)})\right)^{-1} B^T$ and $d(\tilde{u}^{(k)}) = -B\left(D\widetilde{K}(\tilde{u}^{(k)})\right)^{-1}\left(\widetilde{K}(\tilde{u}^{(k)}) - \tilde{f}\right)$. Finally, equation (13) is solved iteratively with a CG or GMRES approach using an additional Dirichlet preconditioner $M_D^{-1}(\tilde{u}^{(k)})$. The Newton-update $\delta\tilde{u}^{(k)}$ is then obtained by solving

$$D\widetilde{K}(\tilde{u}^{(k)})\,\delta\tilde{u}^{(k)} = \widetilde{K}(\tilde{u}^{(k)}) - \tilde{f} - B^T\lambda. \tag{14}$$

The complete Newton-Krylov-FETI-DP algorithm is also presented in Figure 5 (left).

2.2. The parallel application of F and the Dirichlet preconditioner. Solving (13) iteratively, the matrix $F(\tilde{u}^{(k)}) = -B\left(D\widetilde{K}(\tilde{u}^{(k)})\right)^{-1} B^T$ has to be applied to a vector in each iteration. This is indeed a parallelizable and highly scalable routine. We therefore have to consider the structure of the matrix $D\widetilde{K}$ in more details. We omit the evaluation point $\tilde{u}^{(k)}$ here and in the following lines for a better readability. Reordering inner and dual variables and introducing the index set $B := [I, \Delta]$ we can write

$$D\widetilde{K} = \begin{pmatrix} DK_{1_{BB}} & & & D\widetilde{K}_{1_{\Pi B}}^T \\ & \ddots & & \vdots \\ & & DK_{N_{BB}} & D\widetilde{K}_{N_{\Pi B}}^T \\ D\widetilde{K}_{1_{\Pi B}} & \cdots & D\widetilde{K}_{N_{\Pi B}} & D\widetilde{K}_{\Pi\Pi} \end{pmatrix} =: \begin{pmatrix} DK_{BB} & D\widetilde{K}_{\Pi B}^T \\ D\widetilde{K}_{\Pi B} & D\widetilde{K}_{\Pi\Pi} \end{pmatrix}. \tag{15}$$

Since DK_{BB} is block diagonal and the blocks $DK_{i_{BB}}$, $i = 1, ..., N$, can be stored as sequential matrices, e.g., one on each compute core, sequential sparse direct solves can be used to handle applications of DK_{BB}^{-1} to a vector in parallel. Additionally, to perform an application of $F(\tilde{u}^{(k)})$ to a vector, a sparse direct solver has to be used to solve systems involving the coarse matrix $D\widetilde{S}_{\Pi\Pi} := D\widetilde{K}_{\Pi\Pi} - D\widetilde{K}_{\Pi B} DK_{BB}^{-1} D\widetilde{K}_{\Pi B}^T$, which is a Schur-complement in the primal variables of (15). This problem is global and becomes - with a rising size of the coarse space - a parallelization bottleneck. This can be overcome by an inexact coarse solve [26, 28]. For more details on the parallel implementation of an application of $F(\tilde{u}^{(k)})$ and the block factorization of $D\widetilde{K}$, see [26]. In this paper, the considered coarse problems are not too large and we thus always use a sparse direct solver to factorize $D\widetilde{S}_{\Pi\Pi}$. A visualization of DK_{BB} and the coarse space $D\widetilde{S}_{\Pi\Pi}$ can be found in Figure 4 (left, middle).

For the construction of the Dirichlet preconditioner M_D^{-1} we only consider the restriction of the local matrices DK_i, $i = 1, ..., N$ to the inner variables I, i.e., the matrices $DK_{i_{II}}$, $i = 1, ..., N$. A visual representation can again be found in Figure 4 (right). Sparse direct solvers can again be used locally to apply the inverse $DK_{i_{II}}^{-1}$ to a vector. The preconditioner M_D^{-1} is then a weighted sum of the $DK_{i_{II}}^{-1}$, $i = 1, ..., N$. The weights are usually derived from the coefficient functions of the considered partial differential equation.

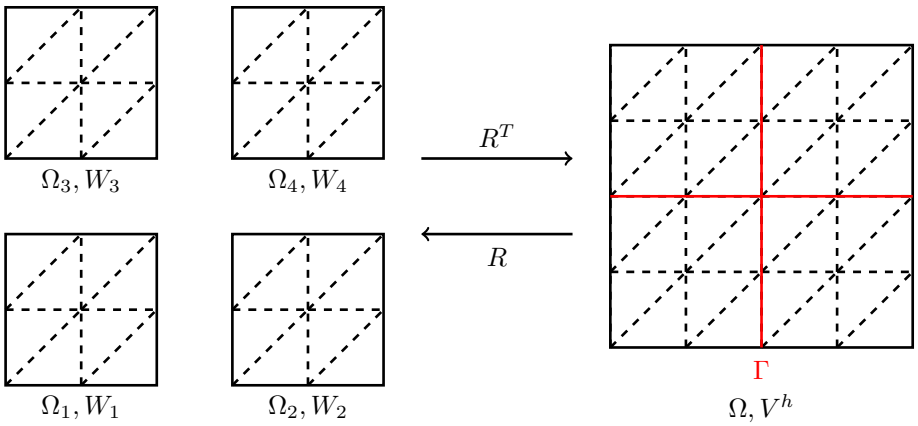


FIG. 2. **Left:** *Decomposition of discretized domain Ω into four subdomains Ω_i, $i = 1, ..., 4$ and corresponding finite element spaces W_i, $i = 1, ..., 4$.* **Right:** *Discretized domain Ω, corresponding finite element space V^h, and interface Γ. The operator R^T acts as a finite element assembly operator on the interface.*

2.3. Nonlinear-FETI-DP. In the nonlinear FETI-DP approach, we first replace the nonlinear equation (6) by a nonlinear saddle point formulation using ideas and operators from linear FETI-DP. With the nonlinear system from (10), coupled in the primal variables, and enforcing the linear jump constraint $B\tilde{u} = 0$ with Lagrange multipliers, we obtain the nonlinear Lagrange function

$$A_{NL}(\tilde{u}, \lambda) := \begin{pmatrix} \widetilde{K}(\tilde{u}) + B^T\lambda - \tilde{f} \\ B\tilde{u} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{16}$$

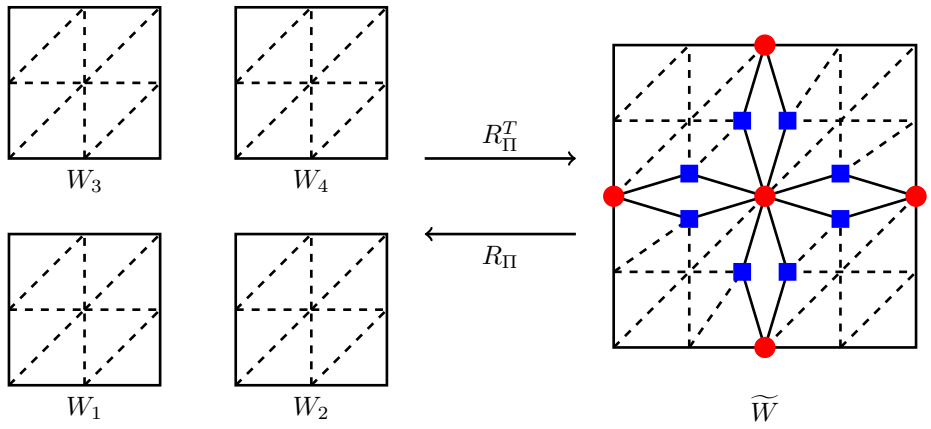


FIG. 3. **Left:** *Decomposition of a discretized domain* Ω *into four subdomains* Ω_i, $i = 1, ..., 4$ *and corresponding finite element spaces* W_i, $i = 1, ..., 4$. **Right:** *Visualization of the primally coupled / assembled space* $\widetilde{W}$. *The subdomains are strongly coupled in the primal constraints (here vertices; red dots; global variables; index* Π*) and still uncoupled in the dual variables (blue squares; local variables; index* Δ*). All remaining variables are considered as inner variables (local variables; index* I*). The operator* R_Π^T *acts as a finite element assembly operator in the primal variables.*

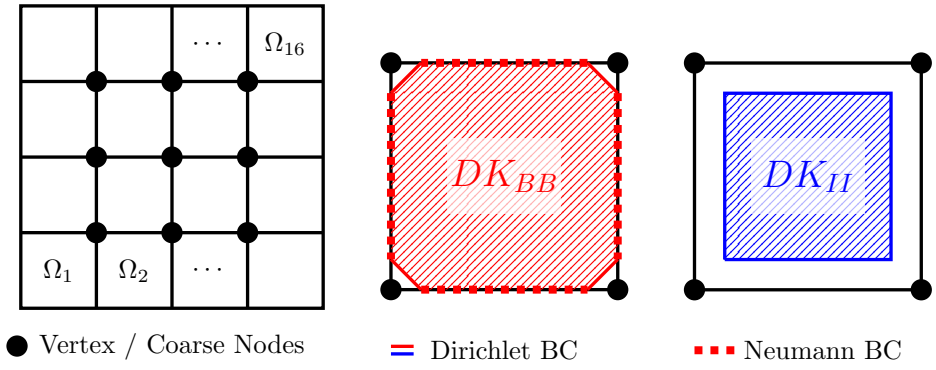


FIG. 4. **Left:** *Domain Decomposition of a computational domain* Ω *into* 16 *subdomains* Ω_i, $i = 1, \ldots, 16$. **Middle:** *Typical local Neumann problem* DK_{BB}^{-1} *occuring on each subdomain (Neumann Boundary Condition everywhere besides in the coarse nodes).* **Right:** *Typical local Dirichlet problem* DK_{II}^{-1} *occuring in the Dirichlet preconditioner* M_D^{-1}.

After applying a nonlinear right-preconditioner $M : \widetilde{W} \times V \to \widetilde{W} \times V$, with $V := \text{range}(B)$, the preconditioned system (5), i.e.,

$$A_{NL}(M(\tilde{u}, \lambda)) = 0,$$

is solved by a Newton-Krylov method. We therefore obtain the solution by the iteration

$$\begin{bmatrix} \tilde{u}^{(k+1)} \\ \lambda^{(k+1)} \end{bmatrix} := \begin{bmatrix} \tilde{u}^{(k)} \\ \lambda^{(k)} \end{bmatrix} - \alpha^{(k)} \begin{bmatrix} \delta\tilde{u}^{(k)} \\ \delta\lambda^{(k)} \end{bmatrix} \tag{17}$$

with the update defined by the linearized system

$$\left(DA_{NL}(M(\tilde{u}^{(k)},\lambda^{(k)}))\cdot DM(\tilde{u}^{(k)},\lambda^{(k)})\right)\begin{bmatrix}\delta\tilde{u}^{(k)}\\ \delta\lambda^{(k)}\end{bmatrix} = A_{NL}(M(\tilde{u}^{(k)},\lambda^{(k)})). \tag{18}$$

Different possible definitions of M are discussed in [26], which all base on a partial nonlinear elimination of variables. It is shown in [26] that (18) can be solved using any linear FETI-DP approach and thus the linear solve in Newton-Krylov-FETI-DP and Nonlinear-FETI-DP only differs by the right hand side. Nonetheless, the preconditioner M has to be applied to $(\tilde{u}^{(k)},\lambda)$ in each iteration, which can make a huge difference. In this paper, we concentrate on Nonlinear-FETI-DP-3 for simplicity, which is defined by a specific choice of M.

2.4. Nonlinear-FETI-DP-3. Using the index set $B := [I,\Delta]$, we rewrite the nonlinear problem (16) as

$$\begin{pmatrix} K_B(u_B,\tilde{u}_\Pi)+B_B^T\lambda - f_B\\ \widetilde{K}_\Pi(u_B,\tilde{u}_\Pi)-\tilde{f}_\Pi\\ B_B u_B\end{pmatrix} = \begin{pmatrix}0\\0\\0\end{pmatrix}. \tag{19}$$

We now decide that the nonlinear preconditioner M is linear in $\tilde{u}_\Pi$ and λ and should eliminate the variables u_B from (19). Therefore, we have $M(\tilde{u},\lambda)^T := (M_{u_B}(u_B,\tilde{u},\lambda)^T, \tilde{u}_\Pi^T, \lambda^T)$ and $M_{u_B}(u_B,\tilde{u},\lambda)$ solves the equation

$$K_B(M_{u_B}(u_B,\tilde{u},\lambda),\tilde{u}_\Pi)+B_B^T\lambda - f_B = 0 \tag{20}$$

To evaluate the preconditioner, (20) has to be solved by Newton's method. Due to the completely decoupled block structure of K_B, the solution of (20) collapses to local Newton iterations, one on each subdomain. The different Newton iterations on different subdomains are independent of each other. This process thus exclusively consists of local work. The nonlinear problems act on the red area marked in Figure 4 (middle) and the linearized systems with the tangential matrices $DK_{i_{BB}}$, $i = 1,...,N$, are solved again by sequential sparse direct solvers. Let us note that the local Newton methods on the individual subdomains might need different numbers of iterations to converge. This introduces the load imbalance mentioned above. With the simpler notation

$$K_B(g_B,\tilde{u}_\Pi)+B_B^T\lambda - f_B = 0 \tag{21}$$

and $g = (g_B,\tilde{u}_\Pi)$ we have $(g,\lambda) = M(\tilde{u},\lambda)$ and the evaluation of the preconditioner can be found in the algorithmic description in Figure 5 (right).

Finally, solving the linear system (18) is equivalent to solving

$$\begin{pmatrix} D\widetilde{K}(g^{(k)}) & B^T\\ B & 0\end{pmatrix}\begin{pmatrix}\delta\tilde{u}^{(k)}\\ \delta\lambda^{(k)}\end{pmatrix} = \begin{pmatrix} rhs_{NL}\\ Bg^{(k)}\end{pmatrix} \tag{22}$$

with

$$rhs_{NL} := \begin{pmatrix}0\\ \widetilde{K}_\Pi(g^{(k)})-\tilde{f}_\Pi\end{pmatrix}.$$

A proof can be found in [26]. Since the left hand sides in (22) and (12) have the same structure, any linear FETI-DP method can be used. As already described for Newton-Krylov-FETI-DP, we reduce the system to the Lagrange multipliers

Init: $\tilde{u}^{(0)} \in \widetilde{W}$
Iterate over k**:**
  If $||\widehat{K}(R_D^T R_\Pi \tilde{u}^{(k)}) - \hat{f}||$ small enough **break;** /* Convergence of Newton-Krylov-FETI-DP; small absolute residual; */
  Solve the linearized system by a Krylov iteration as in a standard linear FETI-DP approach using preconditioner M_D^{-1}:
  $F(\tilde{u}^{(k)})\lambda = d(\tilde{u}^{(k)})$
  Obtain $\delta\tilde{u}^{(k)}$ from λ by solving (14).
  Update: $\tilde{u}^{(k+1)} := \tilde{u}^{(k)} - \delta\tilde{u}^{(k)}$
End Iteration

Init: $\tilde{u}^{(0)} \in \widetilde{W}$, $\lambda^{(0)} \in V$
Iterate over k**:**
  Compute:
  $\left(g^{(k)}, \lambda^{(k)}\right) := M(\tilde{u}^{(k)}, \lambda^{(k)})$
  /* Often requires solution of localized nonlinear problems; inner Newton loop */
  If $||A_{NL}(g^{(k)}, \lambda^{(k)})||$ small enough **break;** /* Convergence of nonlinear FETI-DP; small absolute residual; */
  Solve the linearized system by a Krylov iteration as in a standard linear FETI-DP approach using preconditioner M_D^{-1}:
  $F(g^{(k)})\delta\lambda^{(k)} = d_{NL}(g^{(k)})$
  Obtain $\delta\tilde{u}^{(k)}$ from $\delta\lambda^{(k)}$ by solving (24).
  Update: $\tilde{u}^{(k+1)} := \tilde{u}^{(k)} - \delta\tilde{u}^{(k)}$
  Update: $\lambda^{(k+1)} := \lambda^{(k)} - \delta\lambda u^{(k)}$
End Iteration

FIG. 5. **Left:** *Newton-Krylov-FETI-DP.* **Right:** *Nonlinear-FETI-DP - the part of the code in gray, the evaluation of M, collapses in Nonlinear-FETI-DP-3 into local Newton iterations and the load can be imbalanced; see subsection 2.4.*

$$F(g^{(k)})\delta\lambda^{(k)} = d_{NL}(g^{(k)}) \tag{23}$$

with $d_{NL}(g^{(k)}) = Bg^{(k)} - B\left(D\widetilde{K}(g^{(k)})\right)^{-1} \cdot rhs_{NL}$. Equation (23) is solved iteratively with a CG or GMRES approach using a linear Dirichlet preconditioner $M_D^{-1}(g^{(k)})$ as before. The Newton update $\delta\tilde{u}^{(k)}$ is obtained by solving

$$D\widetilde{K}(g^{(k)})\,\delta\tilde{u}^{(k)} = rhs_{NL} - B^T\delta\lambda^{(k)}. \tag{24}$$

An overview of the algorithm can be found in Figure 5 (right).

3. Nonlinear Model Problems. We choose nonlinear partial differential equations based on the p-Laplace operator with $p \geq 2$ as model problems. These problems are excellent model problems for our purpose, since we can simply create large load imbalances, e.g., by enforcing a linear behavior on certain subdomains by choosing $p = 2$. We basically concentrate on two different setups - the first one has a single nonlinearity in a single subdomain and the second one exhibits a more equal distribution of nonlinear effects. Of course, the first one is designed to emphasize the potential to save energy and the second one is closer to a real application. To describe both model problems in detail, let us first define the scaled p-Laplace operator for $p \geq 2$ by

$$\alpha\Delta_p u := \operatorname{div}(\alpha|\nabla u|^{p-2}\nabla u).$$

Our nonlinear model problem is now defined as

$$\begin{aligned} -\alpha\Delta_p u - \beta\Delta_2 u &= b && \text{in } \Omega \\ u &= 0 && \text{on } \partial\Omega, \end{aligned} \tag{25}$$

where $\alpha, \beta : \Omega \to \mathbb{R}$ are coefficient functions. By variation, applying Green's formula, and a discretization using the finite element space V^h we obtain

$$\int_\Omega \alpha|\nabla u|^{p-2}\nabla u^T\nabla v + \beta\nabla u^T\nabla v \,\mathrm{d}x = \int_\Omega bv \,\mathrm{d}x,\ \forall v \in V^h. \tag{26}$$

A restriction to a subdomain $\Omega_i,\ i = 1, ..., N$, and the corresponding finite element space W_i yields

$$\int_{\Omega_i} \alpha|\nabla u_i|^{p-2}\nabla u_i^T\nabla v_i + \beta\nabla u_i^T\nabla v_i \,\mathrm{d}x = \int_{\Omega_i} bv_i \,\mathrm{d}x,\ \forall v_i \in W_i. \tag{27}$$

Therefore, given a finite element basis $\{\psi_1, \ldots, \psi_n\}$ of V^h, the operators of the global nonlinear problem (as described in (6)) for the given model problem are defined by

$$\begin{aligned}\widehat{K}(\hat{u}) := \Bigg(&\int_\Omega \alpha|\nabla\hat{u}|^{p-2}\ \nabla\hat{u}^T\nabla\psi_1 + \beta\ \nabla\hat{u}^T\nabla\psi_1 \,\mathrm{d}x, ..., \\ &\int_\Omega \alpha|\nabla\hat{u}|^{p-2}\nabla\hat{u}^T\nabla\psi_n + \beta\nabla\hat{u}^T\nabla\psi_n \,\mathrm{d}x\Bigg)^T\end{aligned} \tag{28}$$

and

$$\hat{f} := \left(\int_\Omega b\psi_1 \,\mathrm{d}x, ..., \int_\Omega b\psi_n \,\mathrm{d}x\right)^T.$$

Restriction to a local finite element space W_i, given a finite element basis $\{\varphi_1, \ldots, \varphi_{N_i}\} \subset \{\psi_1, \ldots, \psi_n\}$, yields

$$\begin{aligned}K_i(u_i) := \Bigg(&\int_{\Omega_i} \alpha|\nabla u_i|^{p-2}\ \nabla u_i^T\nabla\varphi_1 + \beta\ \nabla u_i^T\nabla\varphi_1 \,\mathrm{d}x\ , ..., \\ &\int_{\Omega_i} \alpha|\nabla u_i|^{p-2}\nabla u_i^T\nabla\varphi_{N_i} + \beta\nabla u_i^T\nabla\varphi_{N_i} \,\mathrm{d}x\Bigg)^T\end{aligned} \tag{29}$$

and

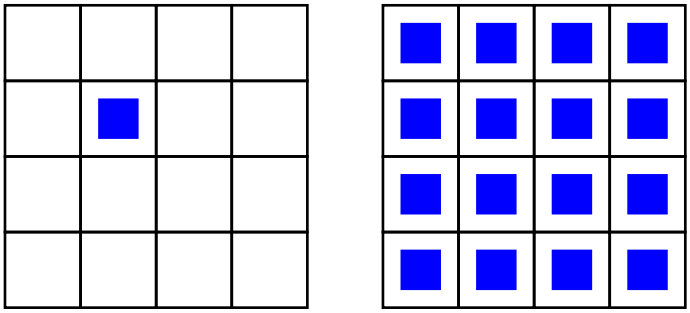
$$f_i := \left(\int_{\Omega_i} b\varphi_1 \,\mathrm{d}x, ..., \int_{\Omega_i} b\varphi_{N_i} \,\mathrm{d}x\right)^T.$$

For the entries of the tangential matrices $DK_i(u_i)$, we obtain

$$\begin{aligned}(DK_i(u_i))_{j,k} := &\int_{\Omega_i} \alpha|\nabla u_i|^{p-2}\nabla\varphi_j^T\nabla\varphi_k \,\mathrm{d}x + \int_{\Omega_i} \beta\nabla\varphi_j^T\nabla\varphi_k \,\mathrm{d}x \\ &+(p-2)\int_{\Omega_i} \alpha|\nabla u_i|^{p-4}\ (\nabla u_i^T\nabla\varphi_j)\ (\nabla u_i^T\nabla\varphi_k) \,\mathrm{d}x\end{aligned} \tag{30}$$

by a direct computation.

For the first configuration with a single nonlinearity, we consider a single inclusion of p-Laplace in a single subdomain and linear 2-Laplace in the remaining domain, i.e., $\alpha = 1$ and $\beta = 0$ in the inclusion and $\alpha = 0$ and $\beta = 1$ in the remaining domain. For the second configuration, where the nonlinearities are distributed equally, we have an equivalent inclusion in each subdomain. In the following, we refer to the first model problem as the problem with a **single inclusion** and to the second one as the problem with **inclusions** or **many inclusions**. For a detailed visualization of α and β in both cases, see Figure 6. Let us remark that we discretize each subdomain into 800×800 pixel and each pixel into two triangular finite elements with linear basis functions. The inclusions always measure 400×400 pixels, i.e., they fill a quarter of a subdomain. We always use $p = 4$ throughout this paper.



$\alpha = 1$ and $\beta = 0$ (nonlinear p-Laplace) in the blue part

$\alpha = 0$ and $\beta = 1$ (linear 2-Laplace) in the remaining part

FIG. 6. **Left:** *Model problem* **single inclusion** *with a single quadratic inclusion of p-Laplace in linear 2-Laplace;* **Right:** *Model problem* **many inclusions** *with quadratic inclusions of p-Laplace in each subdomain.*

4. Reducing the Energy Consumption of Nonlinear-FETI-DP.

4.1. Related Work and Contribution. Reducing energy consumption of HPC applications has gained substantial attraction in the last decade. In the context of MPI several algorithms and runtime systems have been proposed [15, 19, 20, 31–33, 40, 42], which typically follow the same idea: processes exhibiting a high slack time, i. e., time they are blocked with waiting for other processes or communicating, can be clocked down to save energy. This approach is used in [43] for an energy optimized barrier. Furthermore they propose a second optimization where the core waiting in the barrier is shutdown for a certain amount of time. However, they give no details in how this is established and how they obtain the energy measurements. A more fine grained approach is presented in [7], where certain algorithms for barrier synchronization are coupled with dynamic voltage and frequency scaling (DVFS) to save energy during the idle time.

Basically, we combine the idea of shutting down the core in the barrier with speeding up busy cores by accessing the power budget of the "barrier core" to scale up their frequencies. We propose a simple way to shut down cores entering barriers which are known to have sufficiently long wait times. The corresponding energy gains are measured for different nonlinear decomposition methods and the most important contributors to power consumption are investigated.

4.2. Energy Reduction Approach. In the following, we divide the cores into two classes. Cores, where the inner local Newton iteration, i.e., the evaluation of the nonlinear preconditioner M, converges fast, we call *speeder*, whereas the cores which must perform more inner Newton iterations we call *laggers*. All cores are synchronized with a barrier. Speeder cores arrive early and must wait for the laggers to arrive. The residence time of the speeders in the barrier is in the order of seconds for our nonlinear DD method and is visualized in Figure 7 for Newton-Krylov-FETI-DP (top row) and Nonlinear-FETI-DP-3 (bottom row). As expected, the load imbalance for Newton-Krylov-FETI-DP is insignificant for both model problems and there is no potential for a reduction of the energy consumption. In contrast, for Nonlinear-FETI-DP-3, the load imbalance is present for both model problems. Considering a **single inclusion**,

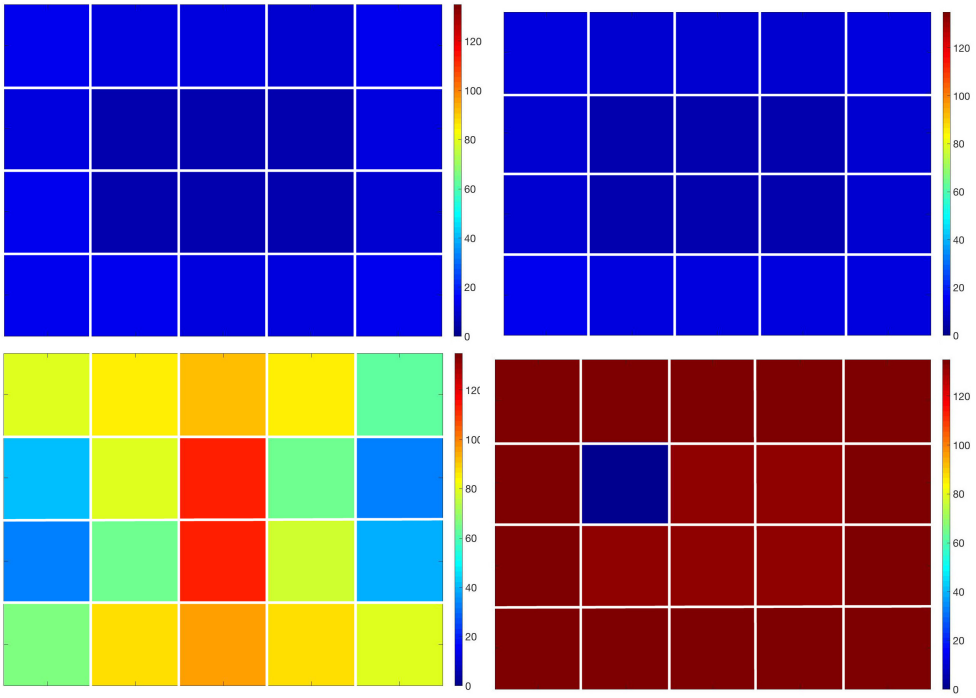


FIG. 7. *Time (given in seconds) of each core/subdomain spent in an MPI_Barrier. Example with 20 cores and subdomains on a single node.* **Top left:** *Many inclusions problem; solved by Newton-Krylov-FETI-DP.* **Top right:** *Single inclusion problem; solved by Newton-Krylov-FETI-DP.* **Bottom left:** *Many inclusions problem; solved by Nonlinear-FETI-DP-3.* **Bottom right:** *Single inclusion problem; solved by Nonlinear-FETI-DP-3.*

there is, as expected, exactly one lagger; see Figure 7 (bottom right).

As already mentioned, dynamic load balancing is currently not feasible. However, we can exploit the load imbalance to reduce the energy consumption of Nonlinear-FETI-DP-3. Here the typical approach for MPI applications is to clock cores down when they are either not in the critical path, i. e., their execution time does not affect the total runtime, or while they remain inside a barrier. This involves a runtime system determining the critical path or programmatically adjusting a core's frequency. Both options are not practical as in most compute cluster production environments adjusting frequencies dynamically for selected cores is not allowed by users. Alternatively, one may rely on the Linux operating system which can adjust a core's frequency automatically depending on the load. To do so a CPU frequency governor must be enabled. This governor is load driven and gracefully adjusts the frequency accordingly. However, we found that cores executing a barrier remained at their highest frequency and were not throttled by the governor. The implementation of the barrier still puts enough pressure on the core to prevent the governor to reduce clock speed.

Instead of reducing a speeder's core frequency, a more effective strategy is to leverage a core's (deep) sleep states. The operating system (OS), e. g., uses these states to save energy when cores are idle. This behavior can implicitly be triggered with current Linux kernels and a corresponding core by calling specific functions like `sleep`, `usleep`, or `nanosleep`. Being in that functions allows the OS to put the core into a deep sleep as long as no other processes are scheduled on this core. These functions

LISTING 1
Non-blocking barrier with test/sleep loop as replacement for `MPI_Barrier`.

```
int flag;
MPI_Request r;

MPI_Ibarrier(comm, &r);

MPI_Test(r, &flag, MPI_STATUS_IGNORE);

while (!flag) {
  usleep(sleep_duration);
  MPI_Test(r, &flag, MPI_STATUS_IGNORE);
}
```

(we choose `usleep`) can be combined with a non-blocking MPI barrier (`MPI_Ibarrier`) as shown in Listing 1 to retain the functionality of the standard barrier (`MPI_Barrier`) and forcing the core into a sleep state. This method also works in situations where the clock frequency of the cores is fixed since DVFS and C-states are orthogonal concepts.

The correct choice of the duration for the sleep (`sleep_duration`) is crucial for several reasons. If the MPI implementation does not rely on progress threads, progress only occurs when an MPI function is called. This is also the case for a barrier, when communication must be performed. With a large sleep duration this can delay the total execution time a core spends inside the barrier and increase the global cost of the barrier. On the other hand if the sleep duration is too small, deep sleep states are not entered. Entering a deep sleep state comes with a certain overhead which is increased the deeper a sleep state is. If the assumed overhead is larger than the sleep duration the desired sleep state is not used. For Nonlinear-FETI-DP-3, our chosen subdomain size, and our model problems we found that 10 ms as sleep duration are short enough to not extend the waiting time in the barrier and long enough to still enter the deepest sleep state. With shorter sleep times, e.g., 5 ms, the deepest sleep state was no longer used exclusively. Note that this behavior might depend on several factors like the application, the application's communication pattern, the job size, the underlying MPI library, the OS, and the processor used.

An important side effect of sleeping cores is that laggers on the same chip have access to a higher fraction of the shared power budget of the chip. If DVFS is enabled they can increase clock speed and reduce their runtime. The more speeders are in deep sleep states, the higher laggers can be clocked and can potentially finish earlier.

4.3. Implementation Remarks and Testbed. All methods are implemented in PETSc [1, 2] based on our nonlinear DD software [24] and using the same basic building blocks. Therefore, the timings and scaling results are comparable. We used PETSc version 3.6.4 and MKL-Pardiso from the MKL (Math Kernel Library) for all sparse direct solves. We refer to [26] for a detailed description of our FETI-DP implementation.

We perform our experiments on the Linux-based Meggie cluster of the RRZE in Erlangen, Germany. All nodes are connected via Intel's high speed 100 GBit Omni-Path network. One compute node consists of two Intel Xeon E5-2630 v4 processors with ten cores each operating at a base frequency of 2.2 GHz. See Table 1 for further details on the compute node. Each processor has a thermal design power (TDP) of 85 W maximum [18]. However, the processor supports several features adjusting its power consumption which are briefly described in the following paragraphs.

Processor name		Intel Xeon E5-2630 v4
Microarchitecture		Broadwell
TDP	[W]	85
Frequency		
Min	[GHz]	1.2
Base	[GHz]	2.2
Cores		10
ISA		AVX2
Sockets		2
L1 cache	[KiB]	32
L2 cache	[KiB]	256
L3 cache	[MiB]	25

TABLE 1
Specifications of a Meggie compute node.

Active cores	1	2	3	4	5	6	7	8	9	10
Maximum Turbo frequency	3.1	3.1	2.9	2.8	2.7	2.6	2.5	2.4	2.4	2.4

TABLE 2
Maximum turbo frequency in GHz depending on the number of active cores for the Intel Xeon E5-2630 v4 processor according to [18]. Reported frequencies specify an upper limit and can be lower depending on e. g., current power consumption or temperature.

The processor supports DVFS which allows for dynamically adjusting each cores frequency individually. A core's frequency can be as low as 1.2 GHz.

Furthermore, Intel's Turbo Boost is supported and enabled. This technique allows for dynamically overclocking a core as long as the chip stays inside its power envelope and does not exceed certain thermal constraints. The maximum clock speed depends on several factors like the temperature of the processor, the current workload, and the number of active cores.

Table 2 lists the maximum Turbo Boost frequencies depending on the number of active cores [18]. Note that these frequencies specify only an upper limit and might be lower because of the previously named reasons.

A core executing AVX workloads, precisely executing 256 bit AVX instructions, gets its frequency dynamically reduced to the so called *AVX frequency*. If, for a certain amount of time, no AVX 256 bit instruction was encountered the core's frequency is raised back again to the *non-AVX frequency*. For the processor model used, it seems that Intel does not provide detailed information about the AVX frequencies. Only [37] lists 1.8 GHz as the base AVX frequency and 3.1 GHz as the highest AVX Turbo Boost frequency.

For advanced energy saving, each core supports (deep) sleep states, also known as *C-states*. A state higher than the normal operating state C0 denotes a core as inactive and thus requires less power. The power consumption decreases with increase in the sleep state level. The processor used supports four states namely C1, C1E, C3, and C6, where the last one denotes the deepest one.

The cluster runs CentOS with Linux kernel 3.10-862. The CPU frequency governor which adapts the clock frequencies is in "conservative mode" on all Meggie nodes. The Intel C/C++ compiler "17.0 update 1" and the Intel MPI library version "2017 update 1" was used. We used the likwid tool suite [41] to measure performance, power

	many inclusions					single inclusion				
	NK	NL, b	NL, b-ts	δ	Δ	NK	NL, b	NL, b-ts	δ	Δ
energy [kJ]	57	44	41	7%	28%	47	29	22	24%	53%
power [W]	136	126	120	5%	12%	136	114	94	18%	31%
runtime [s]	419	348	340	2%	19%	347	251	232	8%	33%

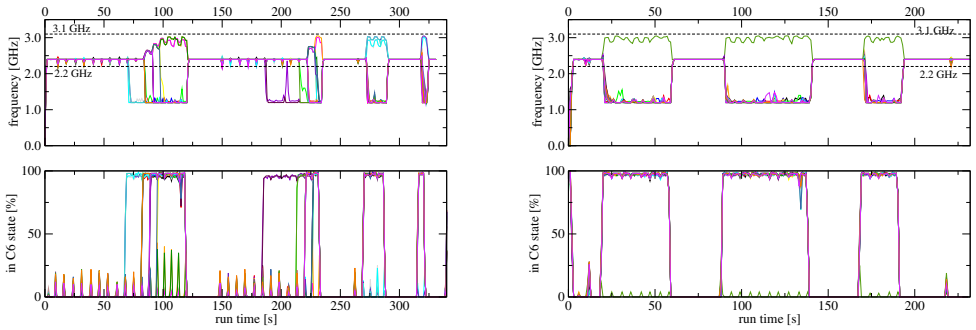
TABLE 3

Energy to solution, power consumption and runtime of Newton-Krylov-FETI-DP (NK) and Nonlinear-FETI-DP-3 (NL) for the many inclusions and the single inclusion problem on one Meggie compute node with standard `MPI_Barrier` *"b" and the non-blocking barrier with test/sleep loop "b-ts". The δ column shows the improvement of the test/sleep loop over the original barrier and the Δ column the improvement over classical Newton-Krylov-FETI-DP.*

and clock speeds. To determine power and energy consumption likwid uses the Intel running average power limit (RAPL) interface which delivers data of high quality [13]. All energy numbers and power consumption measurements include the contributions of the processor chip(s) and the main memory. No other devices are monitored (e.g. power supply or networks).

4.4. Single Node Measurements. Measurements of Newton-Krylov-FETI-DP and Nonlinear-FETI-DP-3 with the standard `MPI_Barrier` "b" and its replacement by a non-blocking barrier with a test/sleep loop "b-ts" for the two model problems are shown in Table 3 for 20 processes on a single Meggie node. Reported energy and power numbers account for the two processors and the memory. As expected, Nonlinear-FETI-DP-3 has a shorter runtime and a lower energy consumption than Newton-Krylov-FETI-DP for both problems. For the **many inclusions** scenario with a moderate load imbalance the two barrier implementations have nearly no impact on the runtime of Nonlinear-FETI-DP-3, but for "b-ts" the energy consumption is reduced by 7 %. Let us also remark that the Nonlinear-FETI-DP-3 implementation with the standard barrier already reduces the consumed energy compared to Newton-Krylov-FETI-DP. This is a combined result of shorter runtime and lower power consumption (see Table 3). Though waiting in the standard barrier (`MPI_Barrier`), a speeder requires less power than a lagger doing computations (see also discussion in section 5 for power consumption analysis of these operations). The more pronounced load imbalance of the **single inclusion** case strongly enhances these effects: Energy savings of up to 53 % and a runtime reduction of 33% compared with Newton-Krylov-FETI-DP are measured. Replacing the barrier by the test/sleep barrier reduces the runtime of Nonlinear-FETI-DP-3 by 8 %.

For a more detailed analysis of Nonlinear-FETI-DP-3, we performed time-resolved measurements. While executing the code, we measure the average core frequency using likwid [41] (averaging interval is 1 second) and we determine the fraction of time spent in the deepest sleep state C6 in each interval (using the `cpu_idle` driver). Figure 8 shows the time-resolved plot for both model problems on a full node of Meggie using 20 cores. Whenever no core is in C6 state (lower panels in Figure 8), all cores achieve the maximum turbo frequency of 2.4 GHz for fully loaded sockets (see upper panel in Figure 8). However, if speeders are entering sleep states they make room for overclocking of the laggers. If some cores' frequencies go down to 1.2 GHz and their fraction of time spent in C6 state is close to 100%, the remaining cores get a boost in clock speed. A nice side effect of the measurements presented in Figure 8 is that one can read out the structure of a typical application of a nonlinear domain decomposition method: An inner iteration, where - one after another - the cores run



(a) many inclusions (b) single inclusion

FIG. 8. *Time-resolved execution of the Nonlinear-FETI-DP-3 algorithm with the test/sleep barrier for the many inclusions (a) and the single inclusion problem (b) for each of the 20 cores of a Meggie (different colors) compute node.* **Top row:** *average frequency during measurement intervals of 1 second;* **Bottom row:** *fraction of the* 1 *second time interval spent in the deepest sleep state C6.*

into some synchronization point followed by a linear solve, where all cores participate with similar effort. For example, for the **many inclusions** case, four outer Newton iterations and linear solves are executed. It is also typical that, with convergence of the outer loop, the inner iterations need fewer steps, since the initial values get closer to the solutions.

Depending on how many cores are already sleeping, i.e., how many cores have finished the inner loop, the processor core frequency is raised to around 3.0 GHz according to the graphs. Even in the case of only a single inclusion in Figure 8b, where only one lagger is present, it seems that the possible 3.1 GHz Turbo Boost frequency is not reached exactly. Several reasons may account for that slight deviation, e.g. the finite length of the averaging interval and the frequent wake ups of the speeders to check for barrier progress.

The measurements in Figure 8 indicate a frequency of 1.2 GHz for cores spending nearly their complete fraction of the 1 second averaging intervals in a deep sleep state. During that time cores are effectively halted and they reduce clock speed down to (almost) zero. The reported value is an artifact of the finite averaging interval being much longer than the sleep states (10 ms). In such scenario likwid reports the clock speed of the core when it wakes up for barrier testing which is done at the lowest possible frequency of 1.2 GHz.

4.5. Multi Node Measurements. For multi node measurements, we employ weak scaling by doubling the number of subdomains in each spatial dimension. Note that the problem with the **single inclusion** keeps only a single inclusion in a single subdomain. Figure 9 shows the total energy consumption and runtime of Newton-Krylov-FETI-DP and Nonlinear-FETI-DP-3 whereas Figure 10 shows the average power per core. As already observed for the single node measurements, Nonlinear-FETI-DP-3 outperforms Newton-Krylov-FETI-DP in terms of time to solution and total energy consumption. Additionally, the weak scaling behavior is superior, not only considering the runtime but also with respect to the power per core; see Figure 10. Of course, the weak scalability is not perfect and the time to solution is slightly increasing. This is a numerical effect and caused by slightly growing numbers of inner iterations. Also the size of the global coarse problem grows proportional to the

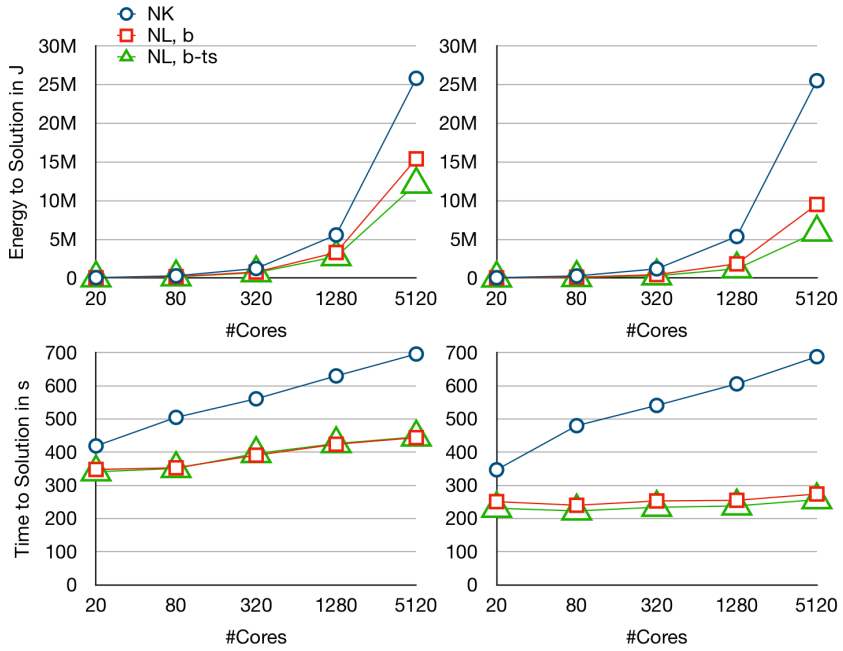


FIG. 9. *Total energy consumption (top panels) and runtime (bottom panels) of Newton-Krylov-FETI-DP and Nonlinear-FETI-DP-3 for both model problems -* **many inclusions** *(left) and* **single inclusion** *(right) - on the Meggie cluster. "NK" is the traditional Newton-Krylov-FETI-DP method where all cores are active at all times. "NL, b" is the Nonlinear-FETI-DP-3 method with standard MPI barriers. "NL, b-ts" is the new implementation making use of test/sleep barriers to save energy and increase the thermal budget of other cores of the same processor.*

number of subdomains, which is a well known bottleneck of the method; see section 2.

Let us note that the power per core is even decreasing during the weak scaling study as with increasing core counts barrier time increases. This characteristic increase of barrier time in overall runtime emphasizes the need for power efficient barriers in large scale computations. Thus, in our study, the effect of the test/sleep barrier is more significant on 256 nodes than on a single node. Here, the energy per core is reduced by 23% for the **many inclusions** case and even 37% for the **single inclusion** example. The runtime for Nonlinear-FETI-DP-3 using the test/sleep barrier is also reduced by 6% for the **single inclusion** problem, since the governor has the opportunity to overclock the single lagger. Let us finally remark that Nonlinear-FETI-DP-3 using a test/sleep barrier can save up to 77 % of energy on 256 nodes compared to Newton-Krylov-FETI-DP (see Figure 9).

4.6. Other Solvers. To illustrate that the concepts introduced here can be carried over to other approaches besides Nonlinear-FETI-DP-3, we briefly present some single node measurements for ASPIN (Additive Schwarz Preconditioned Inexact Newton) [4]. We therefore use the ASPIN implementation available in PETSc [1, 2] and the p-Laplace example provided by PETSc as *ex15.c*; see [3, Section 6.5] for details. Let us remark that in *ex15* finite differences are used instead of finite elements. Written in our notation from section 3, we have $\beta = \varepsilon$ and $\alpha = 1$ on the whole domain. In contrast to [3], we use $p = 4$ and an overlap of ten finite elements.

For a detailed description of ASPIN, we refer to [4]. Let us just remark that ASPIN is a nonlinearly left-preconditioned method and that the nonlinear preconditioner

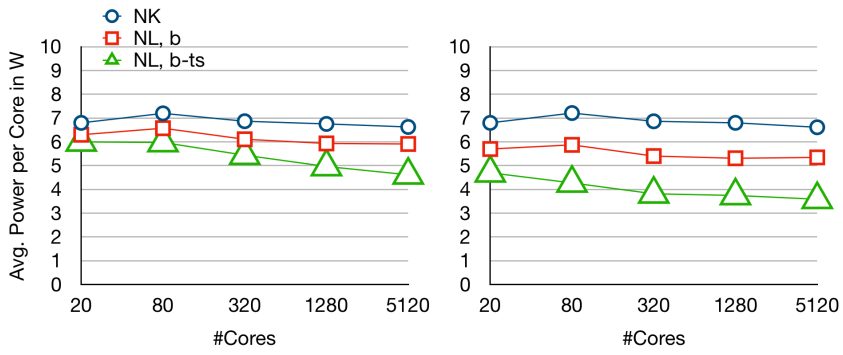


FIG. 10. *Average power consumption per core of Newton-Krylov-FETI-DP and Nonlinear-FETI-DP-3 for two model problems* - **many inclusions** *(left) and* **single inclusion** *(right) - on the Meggie cluster.*

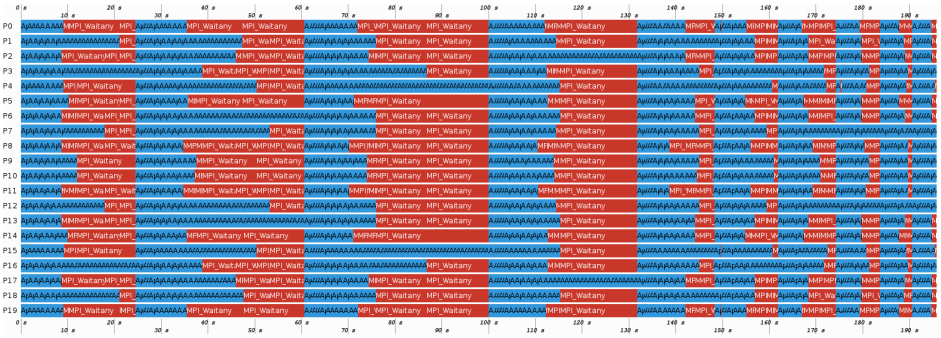


FIG. 11. *Trace of ASPIN obtained with Intel Trace Analyzer on one Meggie node. Blue bars denote time spend in application, whereas red bars denote time spend in MPI; here predominantly MPI_Waitany.*

reduces to independent nonlinear Dirichlet problems on overlapping subdomains.

The ASPIN solver exhibits a load imbalance as shown in Figure 11 when evaluating the nonlinear preconditioner. In contrast to our FETI-DP implementation, where the time was spent in barriers, here the imbalance is visible through long waiting times in `MPI_Waitany` (red bars in Figure 11). We applied the same scheme as for FETI-DP, but instead of replacing the barrier by a test/sleep loop we did this for `MPI_Waitany`. Furthermore, we employ the PMPI interface to transparently replace calls to `MPI_Waitany` by the test/sleep loop.

The energy consumption and runtime for the wait and test/sleep cases are reported in Table 4. As for Nonlinear-FETI-DP-3, test/sleep reduces the energy consumption by 14 % and has no negative impact on the performance.

5. Analysis of basic power contributions. In a final, step we make sense of the power measurements presented in Figure 10 by low level benchmarking and simplistic power modeling. We choose the single inclusion scenario in the limit of large processor counts to identify the relevant power contributions. For the Nonlinear-FETI-DP-3 method this is the extreme case where only one processor is computing throughout while the remaining ones execute the standard barrier or its test/sleep replacement most of the time. Thus, overall power consumption will be determined by executing the standard barrier or the baseline power of the processors being in

		wa	wa-ts	Δ
energy	[kJ]	16	14	-14%
runtime	[s]	136	134	-1%

TABLE 4

Energy consumption and runtime of ASPIN for PETSc example ex15.c on one Meggie compute node with `MPI_Waitany` *"wa" and the test/sleep loop "wa-ts".*

sleep state. For the Newton-Krylov-FETI-DP method time to solution and power consumption will be mostly governed by computation and it can serve as a reference for the power contributions of computations of all methods considered in this paper. To separate these effects we first run several benchmarks on full sockets (10 cores) and report their power consumption and power variation across 500 sockets in Figure 12:

- **barrier**: An artificial micro-application where 10 cores on the first socket are waiting inside an `MPI_Barrier` for the 1st core on the second socket, which is sleeping for several seconds. Power drawn by the first socket is measured only.
- **NK**: A Newton-Krylov-FETI-DP solver used to solve the problem with **many inclusions**. This represents a typical instruction mix for an inexact Newton method using a DD approach as linear solver.

In addition we provide measurements for dense matrix matrix multiplication (using DGEMM from Intel mkl) which is considered to be an upper limit for application power consumption. Note, it is not sufficient to measure a single chip as there can be a

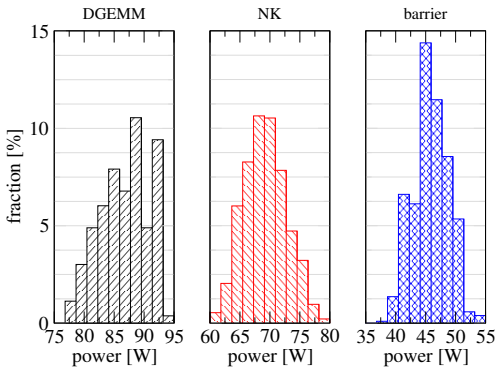


FIG. 12. *Distribution of power consumption of processors and their associated memory from the Meggie cluster under different workloads: DGEMM, Newton-Krylov (NK), and barrier. Note the different scaling on the x axis.*

substantial power variation between different chips of the same processor type running at the same clock speed [17, 39]. Accordingly, for all benchmarks we find substantial power fluctuations across the processor chips in our compute cluster (see Figure 12). The different power levels drawn by these three corner cases directly relate to their hardware utilization as confirmed by likwid measurements for typical hardware utilization metrics such as instruction throughput (IPC) and cache/memory bandwidths. The **barrier** benchmark (IPC $\approx$ 0.4 inst./cycle*; memory bandwidth $<$ 100 MB/s) and DGEMM (IPC $\approx$ 3.3 inst./cycle; cache bandwidths $>$ 50 GB/s) represent the

*This value is still twice as large as the IPC for STREAM benchmark explaining why the governor does not clock down cores as discussed in subsection 4.2

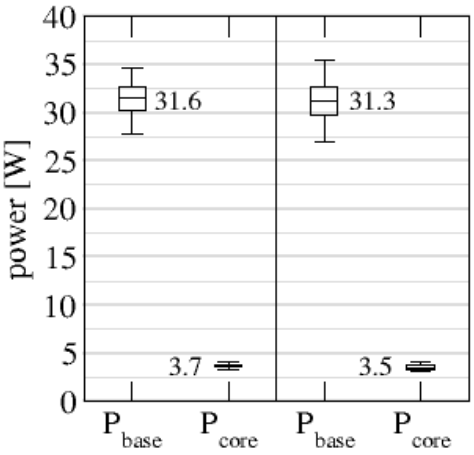


FIG. 13. *Fitted values of base P_{base} and core P_{core} power from single socket measurements on the Meggie cluster for the problem with* **many inclusions** *(left) and a* **single inclusion** *(right) solved with the Newton-Krylov-FETI-DP solver. All values are showing the median.*

extreme cases of hardware utilization and power consumption. The **NK** benchmark is in between drawing a memory bandwidth of approximately 31 GB/s and executing 1.5 instructions per cycle. Extracting the average power per core from Figure 12 and comparing with Figure 10 we find very good agreement between the **NK** benchmark (approx. 7 Watt/core) and the Newton-Krylov-FETI-DP solver as well as between the **barrier** benchmark (approx. 4.5-5 Watt/core) and the Nonlinear-FETI-DP-3 with a standard barrier from MPI. This is clear indication that power consumption of Nonlinear-FETI-DP-3 with prominent load imbalances can be substantially impacted by the MPI barrier.

In a final step we attempt to understand the power level of Nonlinear-FETI-DP-3 with our test/sleep barrier implementation (approx. 3.6 Watt/core; single inclusion in Figure 10). We expect that this barrier implementation should have marginal power contributions as the idle cores are in a deep sleep state. Here, the baseline power of the chip P_{base} should be the dominating factor. At constant clock speed, the total power consumption P_t of a processor chip with c active cores can be approximated by

$$P_t(c) = P_{\text{base}} + cP_{\text{core}}, \tag{31}$$

where P_{core} is the power required to activate an additional core (see e.g. [14]). The unknown values of P_{core} and P_{base} can be determined by fitting Equation (31) to power measurements on single sockets when running the Newton-Krylov-FETI-DP solver with varying the numbers of cores from 1 to 10, i.e. c=1,...,10. Doing these experiments on 500 chips of our Meggie cluster for single and many inclusions benchmark we find $P_{base} \approx 31$ Watt for a full socket (see Figure 13). This is in good agreement with the measured value of 3.6 Watt per core in Figure 10 for the single inclusion benchmark at 5 120 cores. Note, the P_{base} value estimated above is a lower limit for any application running on the system, i.e. 3.1 Watt per core is a lower limit in Figure 10. Executing a barrier (Newton-Krylov-FETI-DP) adds approximately 50% (120%) of dynamic power on top of that (cf. Figure 12).

Our results clearly substantiate the potential high impact of standard barrier implementations on power consumption of applications inhibiting load imbalances. As

future architectures are expected to be more dynamic in terms of power consumption including lower baseline powers, the need for power efficient implementations of solvers and barriers will increase accordingly.

6. Conclusion. We have shown that nonlinear DD methods can not only reduce the energy consumption compared to standard Newton-Krylov-DD approaches by a reduction of time to solution, but also benefit from a better power efficiency. This effect can be even increased by using a nonblocking barrier and actively setting cores in sleep mode. For the example of Nonlinear-FETI-DP-3 and different model problems, energy savings up to 77% can be reached without affecting the runtime. The concepts introduced in this paper can be easily carried over to many nonlinear DD approaches, as, e.g., ASPIN or nonlinear BDDC, and can be combined with approaches to reduce the load imbalance.

REFERENCES

[1] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc users manual*, Tech. Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016, http://www.mcs.anl.gov/petsc.

[2] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Web page*. http://www.mcs.anl.gov/petsc, 2016, http://www.mcs.anl.gov/petsc.

[3] P. R. Brune, M. G. Knepley, B. F. Smith, and X. Tu, *Composing scalable nonlinear algebraic solvers*, SIAM Rev., 57 (2015), pp. 535–565, http://dx.doi.org/10.1137/130936725, https://doi.org/10.1137/130936725.

[4] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200, http://dx.doi.org/10.1137/S106482750037620X, https://doi.org/10.1137/S106482750037620X.

[5] X.-C. Cai, D. E. Keyes, and L. Marcinkowski, *Non-linear additive Schwarz preconditioners and application in computational fluid dynamics*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 1463–1470, http://dx.doi.org/10.1002/fld.404, https://doi.org/10.1002/fld.404. LMS Workshop on Domain Decomposition Methods in Fluid Mechanics (London, 2001).

[6] X.-C. Cai, D. E. Keyes, and D. P. Young, *A nonlinear additive Schwarz preconditioned inexact Newton method for shocked duct flows*, in Domain decomposition methods in science and engineering (Lyon, 2000), Theory Eng. Appl. Comput. Methods, Internat. Center Numer. Methods Eng. (CIMNE), Barcelona, 2002, pp. 345–352.

[7] J. Chen and Y. Dong, *Energy optimization of representative barrier algorithms*, Journal of Central South University, 19 (2012), pp. 2823–2831, http://dx.doi.org/10.1007/s11771-012-1348-z.

[8] V. Dolean, M. J. Gander, W. Kheriji, F. Kwok, and R. Masson, *Nonlinear preconditioning: how to use a nonlinear Schwarz method to precondition Newton's method*, SIAM J. Sci. Comput., 38 (2016), pp. A3357–A3380, http://dx.doi.org/10.1137/15M102887X, https://doi.org/10.1137/15M102887X.

[9] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified FETI method - part i: A faster alternative to the two-level FETI method*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1523–1544.

[10] C. Farhat, M. Lesoinne, and K. Pierson, *A scalable dual-primal domain decomposition method*, Numer. Lin. Alg. Appl., 7 (2000), pp. 687–714.

[11] C. Gross, *A unifying theory for nonlinear additively and multiplicatively preconditioned globalization strategies: Convergence Results and Examples From the field of Nonlinear Elastostatics and Elastodynamics*, PhD thesis, 2009. Deutsche Nationalbibliothek https://www.deutsche-digitale-bibliothek.de/item/PCLVYPVW5OCPUOTIKRKTMSHMFSNWEFPL.

[12] C. Gross and R. Krause, *On the globalization of ASPIN employing trust-region control strategies - convergence analysis and numerical examples*, Tech. Report 2011-03, Inst. Comp. Sci., Universita della Svizzera italiana, 01 2011.

[13] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, *An energy efficiency feature survey of the intel haswell processor*, 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, (2015), pp. 896–904.

[14] G. Hager, J. Treibig, J. Habich, and G. Wellein, *Exploring performance and power properties of modern multi-core chips via simple machine models*, Concurrency and Computation: Practice and Experience, 28, pp. 189–210, http://dx.doi.org/10.1002/cpe.3180, https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3180, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.3180.

[15] C.-h. Hsu and W.-c. Feng, *A power-aware run-time system for high-performance computing*, in Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, SC '05, Washington, DC, USA, 2005, IEEE Computer Society, http://dx.doi.org/10.1109/SC.2005.3.

[16] M. Huber, B. Gmeiner, U. Rüde, and B. Wohlmuth, *Resilience for massively parallel multigrid solvers*, SIAM J. Sci. Comput., 38 (2016), pp. S217–S239, http://dx.doi.org/10.1137/15M1026122, https://doi.org/10.1137/15M1026122.

[17] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, and I. Miyoshi, *Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '15, New York, NY, USA, 2015, ACM, pp. 78:1–78:12, http://dx.doi.org/10.1145/2807591.2807638.

[18] Intel Corp., *Intel Xeon processor E5-2600 v4 product family specification update*, 2016. version: December 2016.

[19] N. Kappiah, V. W. Freeh, and D. K. Lowenthal, *Just in time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs*, in Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, Nov 2005, pp. 33–33, http://dx.doi.org/10.1109/SC.2005.39.

[20] D. J. Kerbyson, A. Vishnu, and K. J. Barker, *Energy templates: Exploiting application information to save energy*, in Proceedings of 2011 IEEE International Conference on Cluster Computing CLUSTER 2011, 2011, http://dx.doi.org/10.1109/CLUSTER.2011.33.

[21] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear bddc methods with inexact solvers*, ETNA. Submitted 2017.

[22] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear FETI-DP and BDDC methods*, SIAM J. Sci. Comput., 36 (2014), pp. A737–A765, http://dx.doi.org/10.1137/130920563, https://doi.org/10.1137/130920563.

[23] A. Klawonn, M. Lanser, and O. Rheinbach, *Nonlinear FETI-DP and BDDC methods*, SIAM J. Sci. Comput., 36 (2014), pp. A737–A765, http://dx.doi.org/10.1137/130920563, https://doi.org/10.1137/130920563.

[24] A. Klawonn, M. Lanser, and O. Rheinbach, *Toward extremely scalable nonlinear domain decomposition methods for elliptic partial differential equations*, SIAM J. Sci. Comput., 37 (2015), pp. C667–C696, http://dx.doi.org/10.1137/140997907, https://doi.org/10.1137/140997907.

[25] A. Klawonn, M. Lanser, and O. Rheinbach, *A highly scalable implementation of inexact nonlinear FETI-DP without sparse direct solvers*, in Numerical mathematics and advanced applications—ENUMATH 2015, vol. 112 of Lect. Notes Comput. Sci. Eng., Springer, [Cham], 2016, pp. 255–264.

[26] A. Klawonn, M. Lanser, O. Rheinbach, and M. Uran, *Nonlinear FETI-DP and BDDC methods: a unified framework and parallel results*, SIAM J. Sci. Comput., 39 (2017), pp. C417–C451, http://dx.doi.org/10.1137/16M1102495, https://doi.org/10.1137/16M1102495.

[27] A. Klawonn and O. Rheinbach, *Robust FETI-DP methods for heterogeneous three dimensional elasticity problems*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1400–1414, http://dx.doi.org/10.1016/j.cma.2006.03.023.

[28] A. Klawonn and O. Rheinbach, *Highly scalable parallel domain decomposition methods with an application to biomechanics*, ZAMM Z. Angew. Math. Mech., 90 (2010), pp. 5–32, http://dx.doi.org/10.1002/zamm.200900329, http://dx.doi.org/10.1002/zamm.200900329.

[29] A. Klawonn and O. B. Widlund, *Dual-Primal FETI Methods for Linear Elasticity*, Comm. Pure Appl. Math., 59 (2006), pp. 1523–1572.

[30] A. Klawonn, O. B. Widlund, and M. Dryja, *Dual-primal FETI methods for three-dimensional elliptic problems with heterogeneous coefficients*, SIAM J. Numerical Analysis, 40 (2002), pp. 159–179.

[31] W. Lavrijsen, C. Iancu, W. de Jong, X. Chen, and K. Schwan, *Exploiting variability for energy optimization of parallel programs*, in Proceedings of the Eleventh European Conference

on Computer Systems, EuroSys '16, New York, NY, USA, 2016, ACM, pp. 9:1–9:16, http://dx.doi.org/10.1145/2901318.2901329, http://doi.acm.org/10.1145/2901318.2901329.

[32] D. Li, B. R. de Supinski, M. Schulz, K. Cameron, and D. S. Nikolopoulos, *Hybrid MPI/OpenMP power-aware computing*, in 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS), April 2010, pp. 1–12, http://dx.doi.org/10.1109/IPDPS.2010.5470463.

[33] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, *Adaptive, transparent CPU scaling algorithms leveraging inter-node MPI communication regions*, Parallel Computing, 37 (2011), pp. 667 – 683, http://dx.doi.org/10.1016/j.parco.2011.07.001.

[34] L. Liu and D. E. Keyes, *Field-split preconditioned inexact newton algorithms*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1388–A1409, http://dx.doi.org/10.1137/140970379, http://dx.doi.org/10.1137/140970379.

[35] L. Liu, D. E. Keyes, and R. Krause, *A note on adaptive nonlinear preconditioning techniques*, SIAM J. Sci. Comput., 40 (2018), pp. A1171–A1186, http://dx.doi.org/10.1137/17M1128502, https://doi.org/10.1137/17M1128502.

[36] L. Marcinkowski and X.-C. Cai, *Parallel performance of some two-level ASPIN algorithms*, in Domain decomposition methods in science and engineering, vol. 40 of Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2005, pp. 639–646, http://dx.doi.org/10.1007/3-540-26825-1_68, https://doi.org/10.1007/3-540-26825-1_68.

[37] Microway, *Detailed specifications of the Intel Xeon E5-2600v4 "Broadwell-EP" processors*. https://www.microway.com/knowledge-center-articles/detailed-specifications-of-the-intel-xeon-e5-2600v4-broadwell-ep-processors/. retrieved 2018-05-06.

[38] C. Negrello, P. Gosselet, and C. Rey, *Nonlinearly preconditioned feti solver for substructured formulations of nonlinear problems*. working paper or preprint, Apr. 2018, https://hal.archives-ouvertes.fr/hal-01774589.

[39] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, *Beyond dvfs: A first look at performance under a hardware-enforced power bound*, in 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, May 2012, pp. 947–953, http://dx.doi.org/10.1109/IPDPSW.2012.116.

[40] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, *Adagio: Making DVS practical for complex HPC applications*, in Proceedings of the 23rd International Conference on Supercomputing, ICS '09, New York, NY, USA, 2009, ACM, pp. 460–469, http://dx.doi.org/10.1145/1542275.1542340.

[41] J. Treibig, G. Hager, and G. Wellein, *Likwid: A lightweight performance-oriented tool suite for x86 multicore environments*, in Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures, San Diego CA, 2010.

[42] A. Vishnu, S. Song, A. Marquez, K. Barker, D. Kerbyson, K. Cameron, and P. Balaji, *Designing energy efficient communication runtime systems: A view from PGAS models*, J. Supercomput., 63 (2013), pp. 691–709, http://dx.doi.org/10.1007/s11227-011-0699-9.

[43] W. Yang and C. Yang, *Exploiting energy saving opportunity of barrier operation in MPI programs*, in 2008 Second Asia International Conference on Modelling Simulation (AMS), May 2008, pp. 144–149, http://dx.doi.org/10.1109/AMS.2008.80.