The Impact of Noise on Krylov Method Performance

Anonymous Author(s)*

ABSTRACT

In this work, we develop a performance model and study the impact of operating system, machine variance, and other factors on Krylov and pipelined Krylov method execution. We perform repeated runs and collect fine-grained data that shows the influence of variability on Krylov and pipelined Krylov methods. To succinctly describe the execution performance, we employ stochastic models based on the distribution of iteration times of each method. We test the models with collected data and suggest ways to improve and expand them in order to make a prior performance estimates.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

Krylov, pipelining, noise, performance model, GMRES, PGMRES

ACM Reference Format:

Anonymous Author(s). 2018. The Impact of Noise on Krylov Method Performance. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Large-scale simulations often use Krylov methods [1] to solve sparse systems of equations in high performance computing (HPC) environments. As we move to more massively parallel computing the latency cost for global communication has skyrocketed [2], degrading the performance of Krylov methods because of the reduction operations used in each iteration. To mitigate the latency associated with global communication, algebraically equivalent pipelined versions of Krylov methods have been introduced [3–8]. Pipelined methods rearrange computations so that it is possible to hide some of the cost of global communication with local computation.

The increasing complexity of HPC computing environments has introduced many sources of run-to-run variability at different levels of HPC systems. For example, operating systems can interrupt processors at any time for its activities, causing detours in computations and degrading performance [11, 12]. Inter-job contention for shared resources such as network bandwidth, routers, and links is another source of variability that can affect application performance [13, 14].

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Highly synchronous applications such as those that use Krylov methods are vulnerable to performance degradation induced by variability at different machine levels. Because each iteration is performed in lockstep, an operating system interrupt that delays a single processor can cause others to be delayed as well. The variable amount of bandwidth available between iterations and runs can degrade Krylov methods because of the use of vector norms and orthogonalizations. It is important to understand and characterize the performance of applications in order to evaluate HPC systems and algorithms. However, developing coherent performance models is difficult in the presence of variability [15, 16] particularly between systems.

Employing stochastic performance models for algorithms that run on HPC systems could reflect a more realistic computing scenario since many detours and sources of variability are unpredictable. Some work has been done using stochastic models to predict performance in HPC environments. For example, [17] studied the impact of different noise distributions on a single collective operation and [18] modeled operating system "jitter" as random variables in order to compute computational slowdown in the presence of noise. In [19], we introduced stochastic performance models that described the performance of Krylov and pipelined Krylov methods in the presence of noise using random variables to model iteration times.

In this work, we are interested in characterizing the performance of Krylov and pipelined Krylov methods on HPC systems in the presence of variability using experimental data and refining the performance model introduced in [19]. We detail the results of experiments performed at the Argonne Leadership Computing Facility (ALCF) and the Swiss National Supercomputing Center and find that, using these results, our stochastic performance models can accurately describe runtimes and suggest a way to perform a priori estimates.

This paper is organized into several sections. Section 2 reviews the performance models introduced in [19] for Kylov and pipelined Krylov methods in the presence of system noise. In Section 3 we examine the results from runs of Krylov and pipelined Krylov methods and refine the performance models in Section 4 based on insights from the experimental data. In Section 5 we employ our stochastic performance models and show that the updated model is in close agreement with reality. In Sections 6 and 7 we present more experimental data and suggest ways to perform a priori performance estimates. Finally, we conclude our work in Section 8.

2 STOCHASTIC MODEL REVIEW

We model a Krylov method as a set of P processes repeatedly performing some local computation and global communication. Because of the synchronizations in each iteration, the processors perform in lockstep so that the total time for each iteration is the time given by the slowest processor. Then the total time T for a

Krylov method is

$$T = \sum_k \max_p T_p^k, \tag{1}$$

where T_p^k is the time for iteration k on process p. Since this work can be interrupted by operating system noise interference, the iteration times T_p^k might fluctuate over steps and processors. To model this nondeterministic behavior, we let the iterates be random variables and ask for the expected total runtime

$$E[T] = \sum_k E[\max_p \mathcal{T}_p^k]. \tag{2}$$

If the iterates $\mathcal{T}_p^k$ are identical and independent of processor (p) and stationary in time (k), then we can compute the expected runtime of a Krylov method using the integral expression

$$E[T] = KP \int_{-\infty}^{\infty} xF(x)^{P-1} f(x)dx, \tag{3}$$

where the random variables $\mathcal{T}_p^k$ are drawn from a distribution with probability density function (pdf) $f(x)$ and cumulative distribution function (cdf) $F(x)$.

Similarly, we model a pipelined Krylov method as a set of P communicating processors. Pipelined methods employ split-phase collectives that do not require global synchronizations. Rather, the collective operation is first initialized, say with `MPI_Ibcast()`, and later finalized with `MPI_Wait()` so that useful work can continue between initialization and finalization. Provided no processor falls too far behind, the total time for a pipelined method T' is the time it take the slowest processor to perform all the given work

$$T' = \max_p \sum_k T_p^k. \tag{4}$$

Because of detours, we do not expect the iteration times to be constant. Again we ask for the expected total run time

$$E[T'] = E[\max_p \sum_k \mathcal{T}_p^k], \tag{5}$$

letting the iterates be random variables drawn from a distribution. If the iterates $\mathcal{T}_p^k$ are identical and independent of process (p) and stationary in time (k), then, in the limit of large K, $\mathcal{T}_p^k \to \mu$ where μ is the mean of the underlying iteration time distribution with pdf $f(x)$ and cdf $F(x)$. So we have an expression to compute the expected total time for a pipelined method

$$E[T'] \to K\mu. \tag{6}$$

In the next section we present fine-grained data collected from runs of Krylov and pipelined Krylov methods. This gives us insight into how detours affect these simulations, which assumptions above are realistic, and how the models might be refined. For a more thorough explanation of the arguments presented in this section, see [19].

3 EXPERIMENTAL RESULTS

Here we present the results from a set of experiments run on the Cray XC40 Theta at the Argonne Leadership Computing Facility. Theta contains Xeon Phi Knights Landing (KNL) compute nodes [20] with 64 CPU processors per node connected by a Cray Aries dragonfly topology [21]. We use the Portable, Extensible Toolkit for Scientific Computation (PETSc) [22, 23] software package to simulate the solution to problems with varying characteristics such as sparsity pattern. One MPI rank per core is used throughout and we boot the nodes in quadrant clustering mode with flat memory mode. We collect iteration times $\mathcal{T}_p^k$ by inserting calls to `MPI_Wtime()` at the beginning and end of each Krylov cycle in PETSc's scalable linear equation solvers (KSP) component's GMRES (generalized minimal residual method) and PGMRES (pipelined GMRES) methods. Since changes and upgrades to HPC machines can affect performance, we note that experiments presented in this paper were performed in autumn 2018.

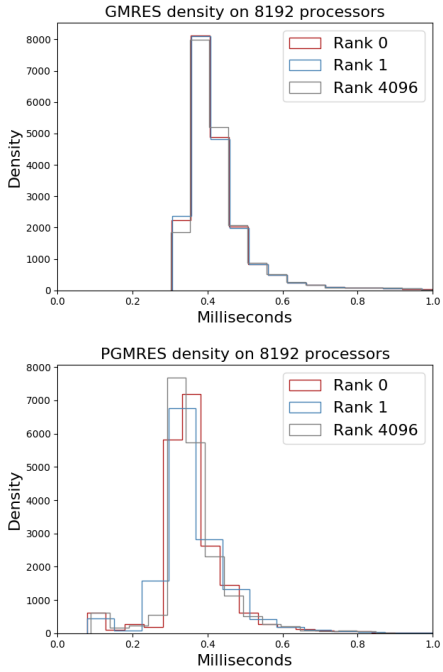

Figure 1: Distribution of iterates $\mathcal{T}_p^k$ for fixed processors.

3.1 One-dimensional Laplacian

PETSc tutorial ex23 is a one-dimensional discretization of the Laplacian, resulting in a simple tridiagonal system. We run ex23 with GMRES and PGMRES on 8192 processors with 10^6 unknowns and a Jacobi preconditioner. Throughout these simulations we force 5000 iterations of the Krylov method.

The stochastic model expressions (3) and (6) place assumptions on the iterates $\mathcal{T}_p^k$. We check them here to examine the actual performance of the algorithms and to justify use of the models. In Figure 1 we graph the distribution of the iterates for select processors identified by their MPI ranks. The distributions look similar between ranks, implying that the iterations might be identical with respect to process. We make use of the two sample Kolmogorov–Smirnov test to check whether two samples (e.g. iterations from rank 0 and rank 1) are drawn from the same underlying distribution. The Kolmogorov–Smirnov test calculates the distance between the empirical distributions of two samples with empirical distribution

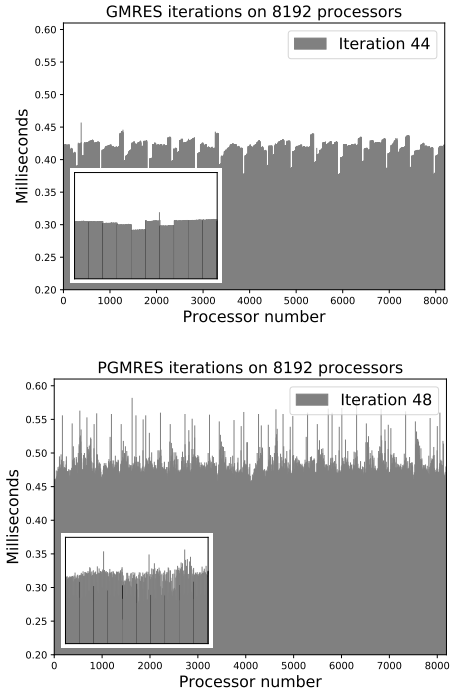

Figure 2: Time for iterations $k = 44$ (GMRES) and $k = 48$ (PGMRES) on process p with inset showing the first 10 nodes.

functions F_1 of size n and F_2 of size m with

$$D = \sup_x |F_1(x) - F_2(x)|.$$

We reject the hypothesis that the samples come from the same distribution with significance level α if

$$D > c(\alpha)\sqrt{\frac{n+m}{nm}}$$

where $c(\alpha)$ is a tabulated value. For GMRES, $n = m = 5000$ and $n = m = 5334$ for PGMRES. We use significance level $\alpha = 0.05$ and SciPy's `stat.ks_2samp` function to calculate the test statistic D. We find that we do not reject that GMRES ranks 0 and 1 come from the same distribution with significance level $\alpha = 0.05$ since $D = 0.002 < 0.024$, the threshold. For PGMRES ranks 0 and 1, $D = 0.088 > 0.023$, so we reject the null hypothesis. On rank 1, we reject none of the $P - 1$ pairs of GMRES ranks and reject 54.5% of PGMRES pairs.

Because it is inefficient to orthogonalize against a very big basis, GMRES methods are often implemented to restart after R iterations. When a Krylov method restarts, the basis for the Krylov space is thrown out and the current solution is used as the initial guess for the next cycle. Since the PETSc default restart value $R = 30$ and each cycle of PGMRES uses two additional iterations to "fill the pipeline," 5334 iterations are actually employed instead of 5000. Some of these are very quick and easily identifiable as the bump of very short iterations in Figure 1 (right).

Figure 2 shows the time for iterations $k = 44$ (GMRES) and $k = 48$ (PGMRES) on process p in order of MPI rank. Iterates are drawn with grey bars and we include a inset graph that shows the iterations on the first ten nodes delineated by black lines. GMRES iterates are nearly constant on a node, each KNL node of 64 processors is clearly visible in Figure 2 (left, inset) so that the iterations are not really independent in p as we claimed above. The variance of iteration times on node 0 (with MPI ranks 0-63) is $6.3 \cdot 10^{-13}$. This could be due to the synchronizations in each iteration of GMRES, forcing processors on a node to perform in lockstep. There is also some periodic behavior over all the nodes. Conversely, PGMRES iterations appear to contain more variability on a node ($1.4 \cdot 10^{-10}$ on the node with MPI ranks 0-63, three orders of magnitude greater than GMRES), but less between them.

Figure 3 is a colormap of the iterates $\mathcal{T}_p^k$ for all iterations and all processors. Graphed are the iterations that perform the "average" amount of work during each GMRES cycle (GMRES iteration 14 mod 30 and PGMRES 16 mod 32) so that each iteration shown performs the same computation. Horizontal lines are highly visible, implying that processors perform similarly within a given iteration. The difference between different iterations is also clear so that the iterates seem to jump around in time without an obvious pattern. There are some groups of iterations (GMRES iterations $k \approx 1700 - 2300$ or PGMRES $k \approx 640 - 740$) that overall take longer than other iterations. This could be explained by some longer operating system interruption.

The derivations in Section 2 made the simplifying assumption that the iterates were stationary in time so that the sum in (2) could become a product in (3). We account for this non-stationary behavior when we refine the performance model in Section 4.

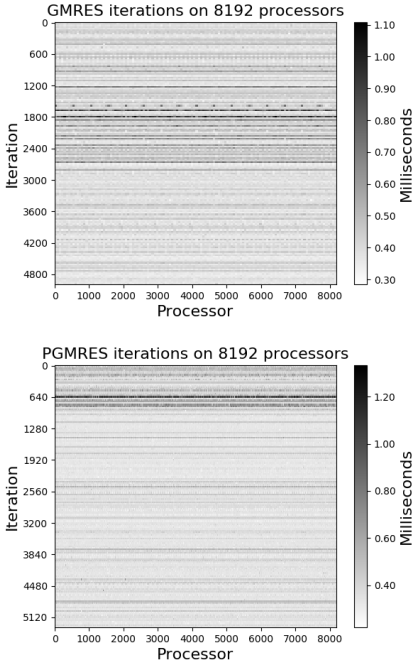

Figure 3: Colormap of iterates $\mathcal{T}_p^k$.

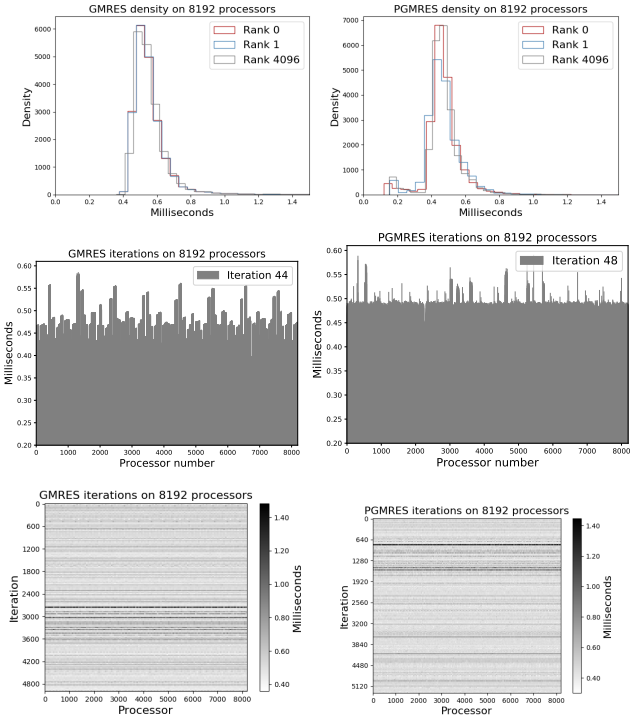

Figure 4: GMRES (left) and PGMRES (right) iterations for PETSc ex48.

3.2 Blatter-Pattyn equations

PETSc ex48 solves the hydrostatic equations for ice sheet flow, where the ice uses a power-law rheology with Glen exponent 3. This generates a much denser system of equations with about 10 times more nonzeros per row than ex23. Again we choose our problem size so that there are 10^6 unknowns, use a Jacobi preconditioner, and force 5000 iterates of the Krylov method. We stop after one nonlinear iteration.

We repeat the analysis from Section 3.1 and find qualitatively similar results, shown in Figure 4. Using the Kolmogorov–Smirnov test again we find that we do not reject the assumption that GMRES iterates from ranks 0 and 1 come from the same distribution since $D = 0.003 < 0.024$ and we reject that PGMRES iterates come from the same distribution since $D = 0.078 > 0.023$. The iterates on a fixed process p look like they could be from the same family of distributions with different parameters as those in the previous subsection.

GMRES iterates again look highly dependent on the node with node 0 variance $4.3 \cdot 10^{-13}$. The variance on PGMRES node 0 is $8.6 \cdot 10^{-10}$, larger than before, and there is more variation between PGMRES nodes which looks periodic. This could be explained by the communication induced by the matrix sparsity pattern. While certain nodes appear to take much longer, the colormap (bottom right) shows that overall processors perform similarly within an iteration. Again, the computation is not stationary in time.

4 NON-STATIONARY PERFORMANCE MODEL

Based on the results from the last section, we move to a non-stationary performance model where the iterates $\mathcal{T}_p^k$ can fluctuate in time. The expected total time for a Krylov method is still given by

$$E[T] = \sum_k E[\max_p \mathcal{T}_p^k]$$

but we relax the claim that the iterates are stationary in time. In iteration k, the random variables $\mathcal{T}_p^k$ are drawn from a distribution with cdf $F_k(x)$ and pdf $f_k(x)$, which can now fluctuate across steps. Then the expected runtime of a Krylov method is given by

$$\widehat{E}[T] = \sum_k P \int_{-\infty}^{\infty} x F_k(x)^{P-1} f_k(x) dx., \tag{7}$$

We call this the "non-stationary" performance model and distinguish between $E[T]$ in (3) and $\widehat{E}[T]$ here.

Similarly for a pipelined method, we use (5) as before and drop the assumption that the iterates are stationary in time, so that

$$\widehat{E}[T'] \to \sum_k \mu_k, \tag{8}$$

where μ_k is the mean of the underlying distribution in iteration k.

The original Krylov model (3) benefits from slight alteration as well. Since the iterates on each node are nearly constant, shown in Figure (2) (left) and Figure 4 (top middle), they are more suitably modeled by $\widehat{P} = P/64$ independent random variables than P independent processors. We make use of this substitution in the next section but it does not seem to effect (7) much.

5 PERFORMANCE MODELING

In this section, we will deploy our performance models and test them against collected data.

5.1 Computing expected runtimes

The expressions for $E[T]$ and $E[T']$ say that the iterates come from some underlying distribution with pdf $f(x)$, cdf $F(x)$, and mean μ. We will use "bulk statistics" with these performance models, taking all iterations for all processors in aggregate, for the underlying distribution and to characterize the iteration times and machine variation.

To find the functions $f(x)$ and $F(x)$ and mean μ, we use Scipy's `stats` package to fit various analytical distributions to our collected data. For a given continuous distribution and data, the `fit` function returns the maximum likelihood estimation for the shape, location, and scale parameters by maximizing a log-likelihood function. We pick the distributions that minimize the sum of squared error between the data and the fit distribution. For the pipelined methods, we disregard the two iterations in each cycle that "fill the pipeline" to avoid distribution fitting complications.

Using these good fitting distributions, we can calculate $E[T]$ and $E[T']$ and compare the expected results to the actual KSPSolve times (the time spent inside the Krylov solver) provided by PETSc's `-log_view` option. For integration in (3) we use Python's quad function and integrate over the bounds of the data.

To employ the updated models $\widehat{E}[T]$ and $\widehat{E}[T']$, we again perform distribution fitting using `scipy`, this time for each iteration to find $f_k(x)$, $F_k(x)$, and mean μ_k. We claim that the runtimes $\mathcal{T}_p^k$ for each iteration k are from the same family with possibly different parameters. Not all iterations resemble the same distribution family, so we use a uniform distribution to model the iterations, shown fitted to GMRES and PGMRES iterations in Figure 6. The distribution parameters vary by iteration. In (7), integration is performed in each iteration again using quad and the calculated expected total times are compared to the KSPSolve time.

Analytical expressions have been used [18] to bound the maximum of a set of random variables using the sample mean μ_X and standard deviation σ_X. Cramer [24, 25] bound the maximum of N identical and independent random variables with

$$E[X_{\max(N)}] \leq \mu_X + \frac{\sigma_X(N-1)}{\sqrt{2N-1}} \tag{9}$$

and Bertsimas [26] bound the maximum of N identical, but not independent random variables by

$$E[X_{\max(N)}] \leq \mu_X + \sigma_X\sqrt{N-1}. \tag{10}$$

We will use these to compare with our Krylov performance models.

5.2 Results

Distribution fitting alone gives some insight into algorithm execution. In Figure 5, we show bulk data from Section 3 with the Johnson SU distribution. Note that there were some other good fitting distributions, including Non-central Student's T. The bulk data show that in a given method, the iterates $\mathcal{T}_p^k$ are mostly clustered and fairly quick with a fewer, longer, giving the distribution a small tail. In general, PGMRES runtimes were faster than GMRES, consistent with faster overall execution in these examples. Ex48 runtimes are shifted to the right of ex23 and more spread out, implying longer iterations with more variation.

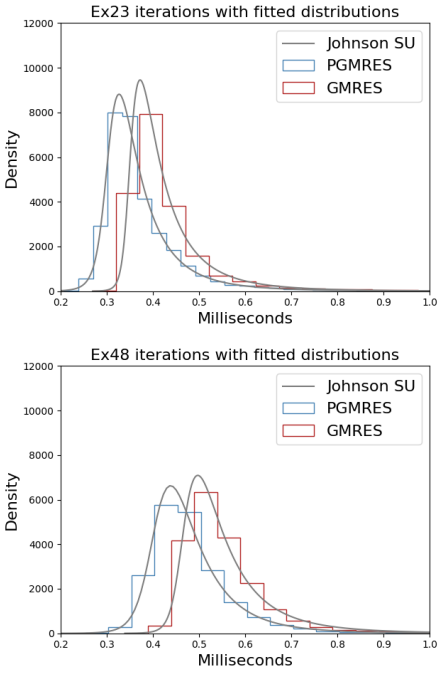

Figure 5: GMRES and PGMRES bulk statistics with fitted distributions for PETSc ex23 (left) and ex48 (right).

Results for the data presented in Section 3 are shown in Table 1 for PETSc ex23 and ex48. We see that the original model for the expected time of a Krylov method $E[T]$ overestimates the actual runtimes and is not a particularly good estimate, nor is any bulk distribution best in all cases. On the other hand, the updated Krylov model $\widehat{E}[T]$ is a big improvement for traditional Krylov methods and makes a reasonable prediction. The analytical bounds are very loose and not particularly helpful. $E[T']$ and $\widehat{E}[T']$ are both good estimates for the pipelined method, suggesting that modeling a method without explicit synchronization is more flexible. In one case, the non-stationary model $\widehat{E}[T']$ is not as good a predictor for PGMRES, but it is more representative of the actual process. The results show that when we have available iteration data, the updated performance models (7) and (8) are good performance predictors and we will use them from here on.

6 PREDICTING RUNTIMES

In the previous section, we showed that our performance models $\widehat{E}[T]$ and $\widehat{E}[T']$ can reasonably predict execution time when we have the time for iteration k on process p for all k and p.

Bulk statistics appear to be enough to succinctly describe the performance of a pipelined method, they are not a sufficient way to describe a traditional Krylov method. Instead, we used non-stationary uniform distributions to model each iteration, shown in Figure 6. Here we see how non-stationary the iterations actually are, in this case they are almost non-overlapping, and how the parameters change over time. That is, for each iteration k, the random variables $\mathcal{T}_p^k$ follow a uniform distribution with pdf f_k and cdf F_k given by

$$f_k(x) = \frac{1}{b_k - a_k}, \quad F_k(x) = \frac{x - a_k}{b_k - a_k}. \tag{11}$$

The uniform distribution can be described with two parameters: the minimum a_k and the span $s_k = b_k - a_k$. We model a Krylov method with K uniform distributions where in each iteration the uniform parameters a_k and s_k are random variables drawn from some distribution. Figure 7 show histograms of the uniform parameters from PETSc ex48 using GMRES and PGMRES with fitted

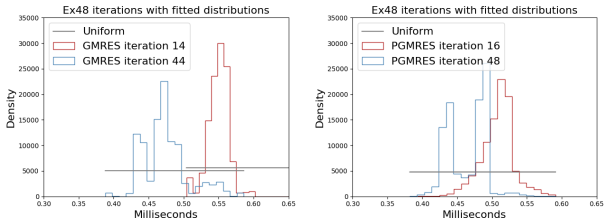

Figure 6: GMRES iterations $k = 14, 44$ and PGMRES $k = 16, 48$ iterations with fitted uniform distributions.

Table 1: Performance model results and bounds for ex23 and ex48 with good fitting distributions.

ex23	KSPSolve	$E_{\text{JohnSU}}[T]$	$E_{\text{NCT}}[T]$	$\widehat{E}_{\text{Unif}}[T]$	Cramer bound	Bertsimas bound
GMRES	2.217	4.831	4.365	2.432	6.35	8.102
PGMRES	2.006	1.865	1.868	1.857		
ex48						
GMRES	2.943	5.716	10.88	3.189	20.59	27.99
PGMRES	2.656	2.413	2.413	2.455		

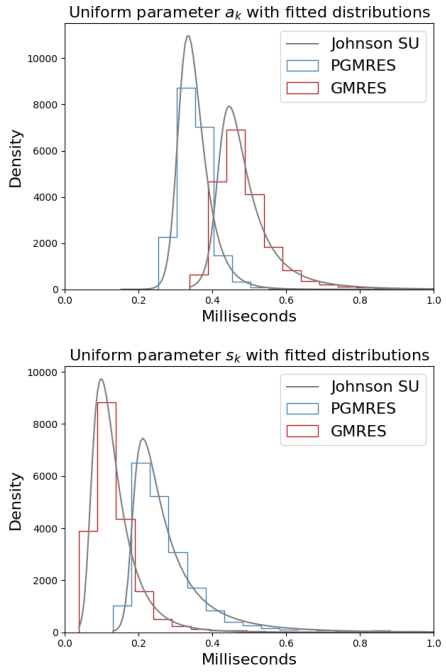

Figure 7: Uniform parameters a_k and s_k from PETSc ex48 with fitted distributions.

Johnson SU distribution. Johnson SU, a transformation of the normal distribution, is described by two shape parameters a and b and location and scale parameters `loc` and `scale` in Scipy.

The location and scale parameters shift the distribution left and right (`loc`) and stretch and shrink horizontally (`scale`) so that standardizing the distribution with `loc` = 0 and `scale` = 1 involves the transformation $y = (x - \texttt{loc})/\texttt{scale}$. Table 2 shows Johnson SU parameters for the uniform distribution parameters from Figure 7 and PETSc ex23. In general, they show that, in a given iteration, the fastest processor is generally faster in a PGMRES run, but the runtimes are more spread out. All Johnson SU distributions are shaped such that they lean left with a tail, some longer than others.

To simulate a Krylov computation, we assume that each iteration time $\mathcal{T}_p^k$ is uniformly distributed with parameters a_k and s_k in iteration k

$$\mathcal{T}_p^k \sim \text{Uniform}(a_k, s_k).$$

We further assume that the parameters a_k and s_k are themselves random variables drawn from a Johnson SU distribution with parameters a, b, `loc`, and `scale`.

We draw random variables a_k and s_k using Scipy's `rvs` function, which generates random variates from a distribution and compute the expected time for iteration k, $\widehat{E}[T_k]$, given by

$$\widehat{E}[T_k] = \widehat{P} \int_{-\infty}^{\infty} x F_k(x)^{\widehat{P}-1} f_k(x) dx. \tag{12}$$

Repeating for K iterations, we get $\widehat{E}[T]$. We simulate a pipelined Krylov computation in the same way, pulling random variables a_k and s_k and computing

$$\widehat{E}[T_k'] = \mu_k \tag{13}$$

where μ_k is the mean of Uniform(a_k, s_k). Table 3 shows the results of this simulation using the Johnson SU parameters from Table 2. The results are good when we have the computed Johnson SU parameters. The challenge in general will be to reason about these parameters so that we can make a priori performance estimates for Krylov and pipelined Krylov methods.

7 EXTENSION TO OTHER ALGORITHMS AND COMPUTING PLATFORMS

In the last section, we saw that we can reasonable simulate the execution time of a Krylov method given Johnson SU parameters to model uniformly distributed iterations. Many factors can influence performance such as algorithm and matrix pattern, as we've seen. In this section, we expand our experiments to study other factors including problem size and computing platform in an effort to test our non-stationary model in other scenarios and suggest ways to perform a priori runtime estimates. Again, these experiments were performed in autumn 2018.

7.1 Scaling experiments

Our experiments so far have been performed on 8192 processors with 10^6 unknowns. To see how changing processor count and problem size affect the Krylov method performance and underlying uniform parameters, we run strong- and static-scaling experiments. Figures 8 and 9 show the Johnson SU distributions for the uniform parameters a_k and s_k for strong- and static-scaling experiments.

With strong-scaling experiments, we see how varying the processor count affects Krylov method performance for a fixed problem. We repeat runs of ex48 with 10^6 unknowns on $P = 1024 - 8192$ processors. We see that, for a fixed number of unknowns, as we decrease problem size the Johnson SU distribution shifts to the right and spreads out for both uniform parameters since each processor does more work and is more likely to experience a detour. Similarly,

Table 2: Johnson SU parameters for uniform parameters a_k and s_k.

	Uniform a_k				Uniform s_k			
ex23	a	b	loc	scale	a	b	loc	scale
GMRES	-5.86e-01	3.35	3.97e-04	1.07e-21	-7.40e-01	3.21	8.06e-04	1.86e-23
PGMRES	2.84	6.74	3.10e-04	2.26e-19	-6.18e-02	2.42	2.21e-03	6.47e-19
ex48								
GMRES	6.74e-01	2.09	2.22e-02	2.24e-24	6.69e-01	2.10	1.71e-03	9.23e-26
PGMRES	-6.02e-01	3.34	4.07e-04	3.05e-23	1.24	1.84	1.36e-02	1.66e-18

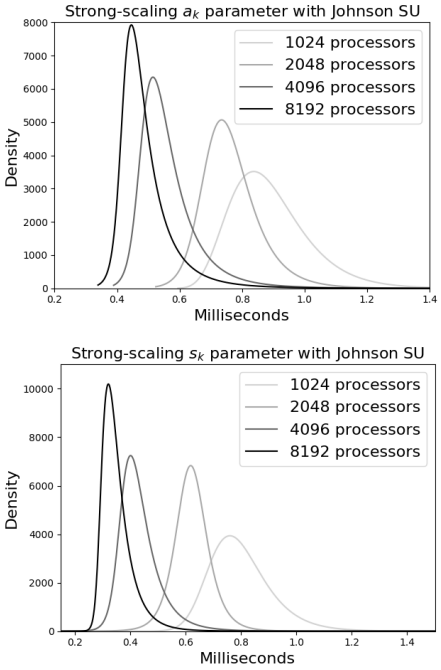

Figure 8: GMRES ex48 strong-scaling results on $P = 1024 - 8192$ processors.

we perform static-scaling results where we keep the number of processors fixed at $P = 8192$ and vary the number of unknowns to see the effect of computation on underlying iteration distributions. Static-scaling changes exaggerate what we see for strong-scaling.

7.2 BCGS

We solve PETSc ex23, the one-dimensional Laplacian problem, using the Stabalized biconjugate gradient method (BCGS) and a pipelined version (PIPEBCGS) with 5000 linear iterations and 10^6 unknowns on Theta.

Table 3: Actual and simulated runtimes in seconds on 8192 processors with 10^6 unknowns.

ex23	GMRES	PGMRES	ex48	GMRES	PGMRES
KSPSolve	2.217	2.006	KSPSolve	2.943	2.656
Simulated	2.416	1.908	Simulated	3.14	2.482

Many aspects of the BCGS and PIPEBCGS computations were consistent with GMRES and PGMRES, such as non-stationary iterates and nearly constant BCGS runtimes on a given node of Theta. Figure 10 shows uniform parameter a_k and s_k histograms with GMRES and PGMRES for comparison. Although with different parameters, the BCGS and PIPEBCGS distributions look like they can be modeled by Johnson SU distribution family, suggesting that our performance model is applicable to Krylov methods outside of GMRES methods. GMRES and PGMRES outperform BCGS and PIPEBCGS and have quicker minimum iteration times (Johnson SU distributions shifted to the left for uniform parameter a_k), but runtimes are tightly grouped. Performance results for BCGS and PIPEBCGS using our non-stationary stochastic models are in shown Table 4.

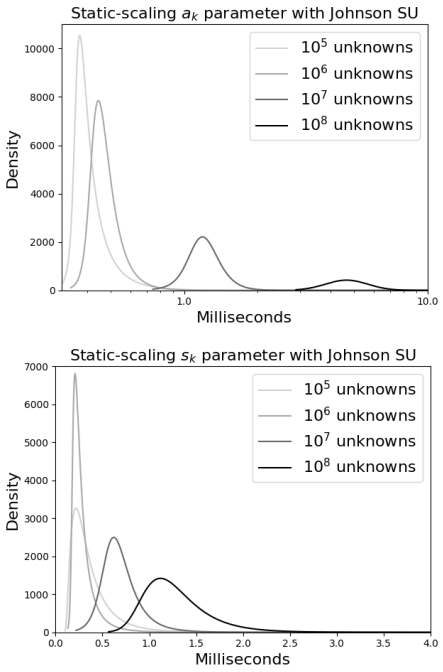

Figure 9: GMRES ex48 static-scaling results for $10^5 - 10^8$ unknowns.

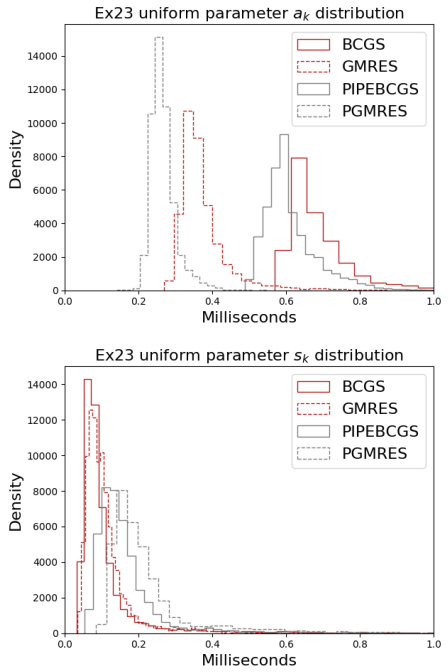

Figure 10: BCGS and PIPEBCGS uniform parameters compared to GMRES and PGMRES.

7.3 Piz Daint

We repeat experiments on the Cray XC40 Piz Daint supercomputer at the Swiss National Computing Center. Piz Daint and Theta both employ a Aries Dragonfly interconnect but Piz Daint contains Intel Xeon E5 Haswell processors [27] on compute nodes, so the differences we see are likely due to different processors or the shared use of a communication network. Figure 11 shows runs of ex48 with GMRES and PGMRES on 8192 processors and 10^6 unknowns. In a given iteration, the quickest processor can be much faster on Piz Daint than Theta, but both uniform parameters contain much more variation. Non-stationary performance model results from Piz Daint are in shown Table 4.

7.4 Mira

We repeat ex48 experiments on Mira, an IBM Blue Gene/Q [28] at Argonne's ALCF. We expect a marked performance difference on Mira since nodes on Mira are connected by a 5D Torus Network with hardware to assist collective functions, where nodes on Theta are connected by a Cray Aries Network where neighboring workloads share network routes, increasing performance variability [29]. Each Blue Gene/Q node also contains a redundant processor that can soak up operating system interrupts, similar to Cray's core specialization feature where one core per node is dedicated to handling OS operations, which reduces core variability [14].

Figure 12 shows histograms of iterations from difference places in the GMRES cycle as well as a bulk histogram for all iterations from all processors, which can be thought of as the average performance. Each curve has been normalized to represent a distribution. GMRES iterations on Theta at different places in the cycle appear similar to the average. They are noisy enough that our simulation from Section 6 provided reasonable results even though we drew uniform parameters a_k and s_k from the same Johnson SU distributions regardless of k in the GMRES cycle. With Mira's quiet network, the GMRES iterates are much more distinct suggesting that a refined model that accounts for the amount of work done per iteration would be needed for a priori estimates. Futhermore, we see a jump in the time between iterations $k \mod 30 = 13$ and $k \mod 30 = 14$ probably due to cache effects and using more storage throughout a GMRES cycle. Similar results were found for PGMRES and performance results are in Table 4.

8 CONCLUSION AND FUTURE WORK

The goal of this work was to gather fine-grained iteration data from runs of Krylov and pipelined Krylov methods to develop a performance model and study the impact of machine interference

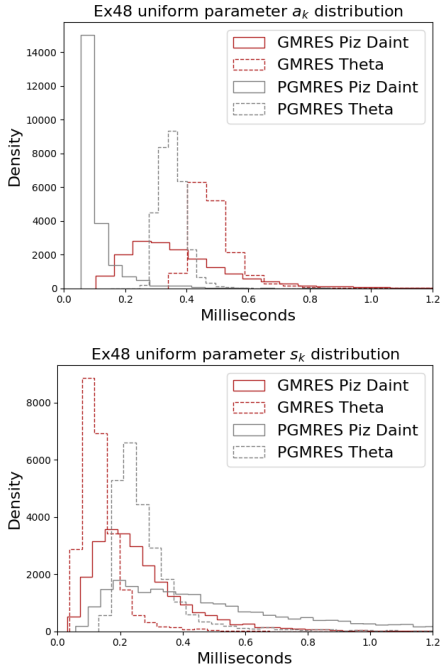

Figure 11: GMRES and PGMRES on Theta and Piz Daint.

Table 4: Non-stationary performance model results for ex23 and ex48 for extended experiments.

	Theta			
ex23	BCGS	PIPEBCGS		
KSPSolve	3.957	3.53		
$\widehat{E}_{\text{Unif}}[T]$	4.227	3.512		
	Piz Daint		Mira	
ex48	GMRES	PGMRES	GMRES	PGMRES
KSPSolve	3.029	2.642	3.026	2.804
$\widehat{E}_{\text{Unif}}[T]$	3.541	2.473	3.189	2.455

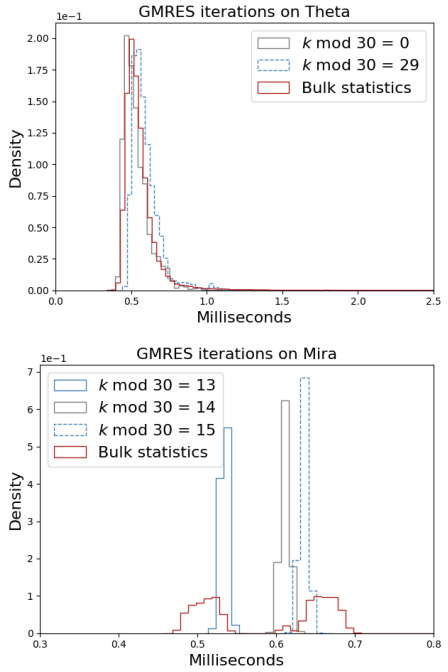

Figure 12: GMRES ex48 iteration density and Theta and Mira.

on these methods. We used the data to develop and test stochastic performance models, which are in good agreement with reality. Variability during computation suggests that performance is not deterministic and could be better explained in a stochastic setting. With descriptive enough statistics we were also able to make a priori performance estimates. In future work, nondeterministic performance models can be derived for other parallel algorithms where unpredictable system interference could impact performance. Insights from performance models can also be used to guide the development of new algorithms, particularly those we expect to be running in less predictable computing environments, such as heavily loaded machines or loosely coupled networks used in cloud computing or those with shared resources.

REFERENCES

[1] Saad Y. Iterative Methods for Sparse Linear Systems. Boston: PWS Publishing Company; 1996.

[2] Dongarra J, Luszczek P, et al., HPC Challenge; 2015. http://icl.cs.utk.edu/hpcc/hpcc_results_lat_band.cgi?display=opt.

[3] Chronopoulos AT, Gear CW. *s*-step iterative methods for symmetric linear systems. Journal of Computational and Applied Mathematics 1989;25:153–168.

[4] Ghysels P, Ashby TJ, Meerbergen K, Vanroose W. Hiding global communication latency in the GMRES algorithm on massively parallel machines. Leuven, Belgium: Intel Exascience Lab; 2012.

[5] Ghysels P, Vanroose W. Hiding Global Synchronization Latency in the Preconditioned Conjugate Gradient Algorithm. Parallel Computing 2014;40(7):224–238. http://www.sciencedirect.com/science/article/pii/S0167819113000719, 7th Workshop on Parallel Matrix Algorithms and Applications.

[6] Strzodka R, Göddeke D. Pipelined Mixed Precision Algorithms on FPGAs for Fast and Accurate PDE Solvers from Low Precision Components. In: Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines IEEE Computer Society; 2006. p. 259–270. FCCM '06.

[7] Sturler ED, van der Vorst HA. Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. Applied Numerical Mathematics 1995;18:441–459.

[8] Jacques T, Nicolas L, Vollaire C. Electromagnetic Scattering with the Boundary Integral Method on MIMD Systems. In: High-Performance Computing and Networking, vol. 1593 of Lecture Notes in Computer Science Springer; 1999.p. 1025–1031.

[9] Cools S. Numerical stability analysis of the class of communication hiding pipelined Conjugate Gradient methods. arXiv preprint arXiv:180402962 2018;.

[10] Carson E, Rozložník M, Strakoš Z, Tichý P, Tůma M. The Numerical Stability Analysis of Pipelined Conjugate Gradient Methods: Historical Context and Methodology. SIAM Journal on Scientific Computing 2018;40(5):A3549–A3580. https://doi.org/10.1137/16M1103361.

[11] Hoefler T, Schneider T, Lumsdaine A. Characterizing the influence of system noise on large-scale applications by simulation. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis IEEE Computer Society; 2010. p. 1–11.

[12] Ferreira KB, Bridges P, Brightwell R. Characterizing Application Sensitivity to OS Interference Using Kernel-level Noise Injection. In: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing SC '08, Piscataway, NJ: IEEE Press; 2008. p. 19:1–19:12. http://dl.acm.org/citation.cfm?id=1413370.1413390.

[13] Parker S, Morozov V, Chunduri S, Harms K, Knight C, Kumaran K. Early Evaluation of the Cray XC40 Xeon Phi System Theta at Argonne. Cray User Group 2017 proceedings 2017;.

[14] Chunduri S, Harms K, Parker S, Morozov V, Oshin S, Cherukuri N, et al. Run-to-run variability on Xeon Phi based Cray XC systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis ACM; 2017. p. 52.

[15] Beckman P, Iskra K, Yoshii K, Coghlan S. The influence of operating systems on the performance of collective operations at extreme scale. In: Cluster Computing, 2006 IEEE International Conference on IEEE; 2006. p. 1–12.

[16] Hoefler T, Lumsdaine A, Rehm W. Implementation and Performance Analysis of Non-Blocking Collective Operations for MPI. In: Proceedings of the 2007 International Conference on High Performance Computing, Networking, Storage and Analysis, SC07 IEEE Computer Society/ACM; 2007. .

[17] Agarwal S, Garg R, Vishnoi NK. The impact of noise on the scaling of collectives: A theoretical approach. In: International Conference on High-Performance Computing Springer; 2005. p. 280–289.

[18] Seelam S, Fong L, Tantawi A, Lewars J, Divirgilio J, Gildea K. Extreme scale computing: Modeling the impact of system noise in multicore clustered systems. In: Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on IEEE; 2010. p. 1–12.

[19] Morgan H, Knepley MG, Sanan P, Scott LR. A stochastic performance model for pipelined Krylov methods. Concurrency and Computation: Practice and Experience 2016;28(18):4532–4542.

[20] Sodani A. Knights landing (knl): 2nd generation intel® xeon phi processor. In: Hot Chips 27 Symposium (HCS), 2015 IEEE IEEE; 2015. p. 1–24.

[21] Alverson B, Froese E, Kaplan L, Roweth D. Cray XC series network. Cray Inc, White Paper WP-Aries01-1112 2012;.

[22] Balay S, Abhyankar S, Adams M, Brown J, Brune P, Buschelman K, et al. Petsc users manual revision 3.8. Argonne National Lab.(ANL), Argonne, IL (United States); 2017.

[23] Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley MG, McInnes LC, et al., PETSc web page; 2001.

[24] Cramér H. Mathematical methods of statistics (PMS-9), vol. 9. Princeton university press; 2016.

[25] David HA, Nagaraja HN. Order statistics. Encyclopedia of Statistical Sciences 2004;9.

[26] Bertsimas D, Natarajan K, Teo CP. Tight bounds on expected order statistics. Probability in the Engineering and Informational Sciences 2006;20(4):667–686.

[27] Hammarlund P, Martinez AJ, Bajwa AA, Hill DL, Hallnor E, Jiang H, et al. Haswell: The fourth-generation intel core processor. IEEE Micro 2014;34(2):6–20.

[28] Kumaran K. Introduction to Mira. In: Code for Q Workshop; 2016. .

[29] Groves T, Gu Y, Wright NJ. Understanding Performance Variability on the Aries Dragonfly Network. In: Cluster Computing (CLUSTER), 2017 IEEE International Conference on IEEE; 2017. p. 809–813.