Understanding performance variability in standard and pipelined parallel Krylov solvers

Hannah Morgan[1], Patrick Sanan[2], Matthew Knepley[3], Richard Mills[1]

[1] *Argonne National Laboratory*, [2] *ETH Zurich*,
[3] *University at Buffalo*

June 6, 2019

Outline

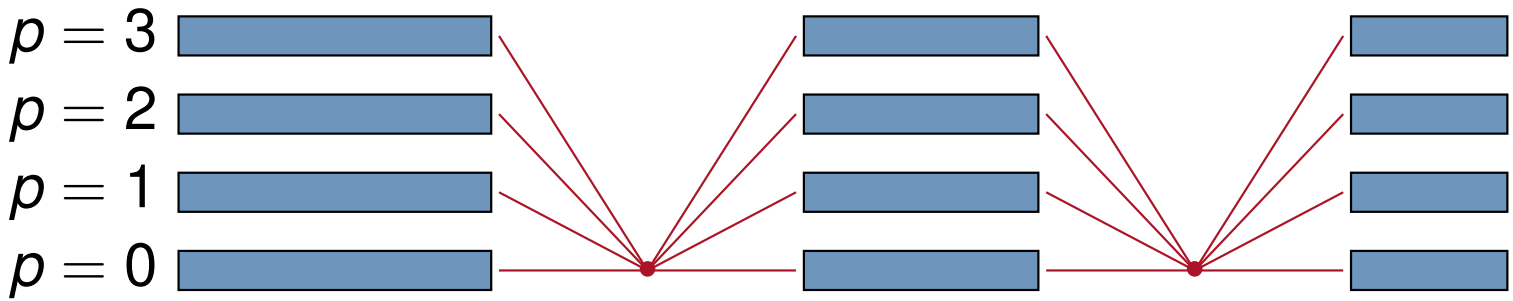
- Introduction and motivation
- Stationary performance model
- Non-stationary performance model
- Performance prediction
- Extensions (more Krylov methods, computing platforms, and GPUs)

Krylov methods

Krylov subspace methods are iterative methods that solve systems of equations $Ax = b$ by looking for solutions in a Krylov subspace of size m

$$\mathcal{K}_m = \text{span}\{b, Ab, A^2b, \ldots, A^{m-1}b\}.$$

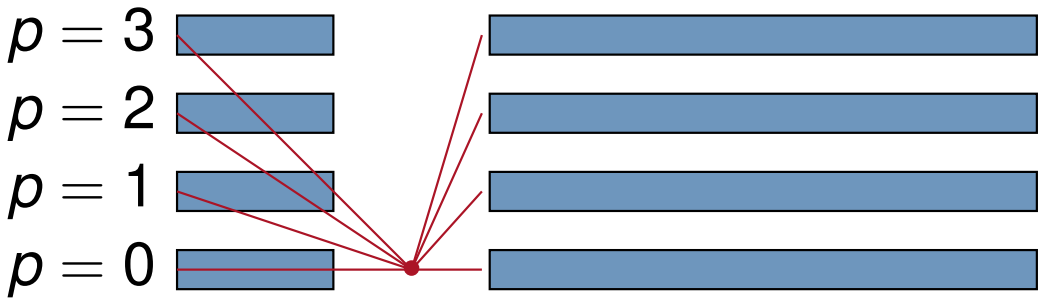
A parallel version of GMRES has two global communication phases in each iteration.



Argonne NATIONAL LABORATORY

Pipelined Krylov methods

A pipelined version of GMRES has only one and allows local computation to overlap with communication.



Images and KSPPGMRES (p^1-GMRES) from Ghysels, Ashby, Meerbergen, and Vanroose.

Motivation

We obtained data from repeated runs of PETSc KSP tutorial ex23 on 8192 processors on the Cray XC30 supercomputer Piz Daint.

We have 12 runs of GMRES and PGMRES including the runtimes (in seconds) below.

$$\text{GMRES:} \quad [0.661, \ldots, 0.982, \ldots, 1.074]$$

$$\text{PGMRES:} \quad [0.464, \ldots, 0.611, \ldots, 0.769]$$

How should we report the runtimes? We could use the left or right endpoints, or the mean of the data.

Motivation

We also have 20 runs of CG and PIPECG including the runtimes below.

CG: $[0.605, \ldots, 0.883, \ldots, 1.605]$

PIPECG: $[0.554, \ldots, 0.682, \ldots, 1.076, 1.695]$

Again, it isn't clear what we should say about the performance.

Deterministic performance model

We can model a Krylov method as a set of P communicating processes who perform a calculation consisting of local computation (blue) and periodic global synchronizations with, perhaps, waiting (red).

The total running time T for K iterations is

$$T = \sum_k \max_p T_p^k,$$

where T_p^k is the time for iteration k on process p.

Argonne NATIONAL LABORATORY

Deterministic performance model

We remove the synchronizations to model a pipelined method.

In this case, the total running time T' is

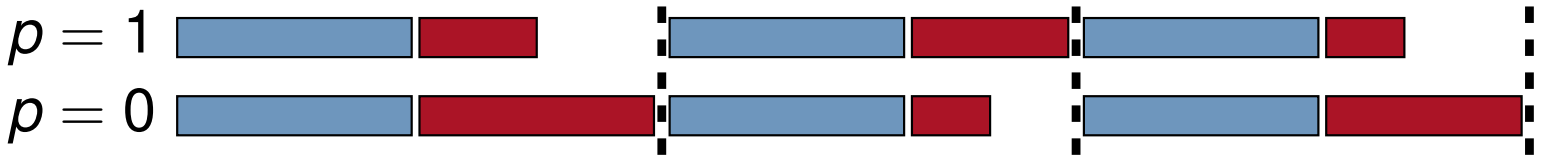
$$T' = \max_p \sum_k T_p^k.$$

Realistic computing scenario

It is not always reasonable to expect predictable delays since there are many sources of variability in HPC environments (e.g. operating system interrupts or inter-job contention for shared resources). Instead, we will turn to a stochastic performance model where workloads are seen in a statistical sense.

Stochastic performance model

Again, we model a Krylov method as a set of P communicating processes.



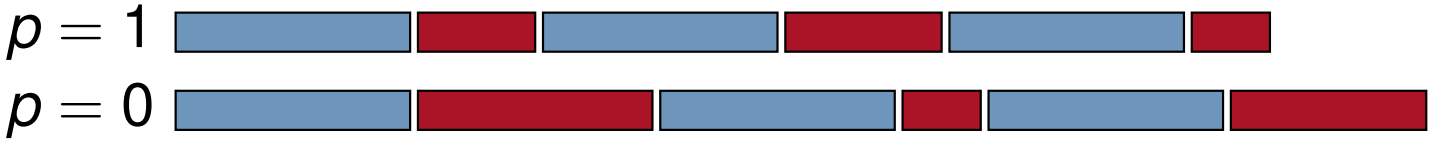
Now, the total running time T for K iterations is

$$T = \sum_k \max_p \mathcal{T}_p^k,$$

where $\mathcal{T}_p^k$ is the random variable representing the time for iteration k on process p.

Stochastic performance model

We remove the synchronizations to model a pipelined method.



Again, the total running time T' is

$$T' = \max_p \sum_k \mathcal{T}_p^k.$$

Stochastic performance model

We let the time for iteration k on processor p be the random variable $\mathcal{T}_p^k$ and ask for the expected runtime of a Krylov method

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

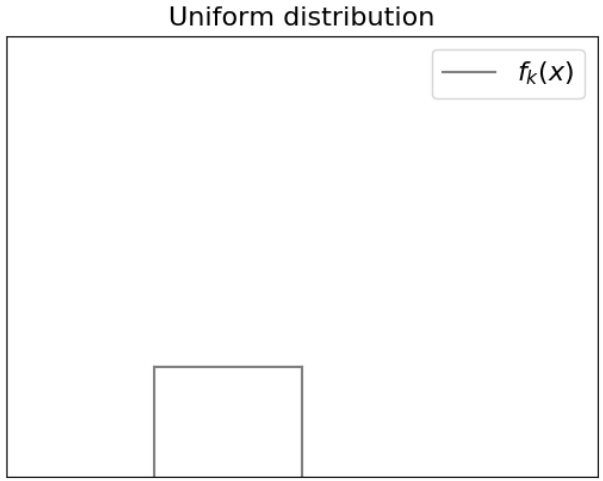
and of a pipelined method

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

We want to study the iteration times $\mathcal{T}_p^k$ so that we can make reasonable performance estimates for Krylov and pipelined Krylov methods.

Stationary performance model

First, we assume that the iteration times $\mathcal{T}_p^k$ are stationary in time. That is, we assume that $\mathcal{T}_p^k$ follow the same distribution for all k.



For example, the iteration times might be uniformly distributed with parameters $(1, 1)$ for all k.

Stationary performance model

Assuming that runtimes are identical and independent of process p and stationary in step k, we get

$$E[T] = \sum_k E[\max_p \mathcal{T}_p^k] = PK \int_{-\infty}^{\infty} xF(x)^{P-1} f(x) dx$$

for a Krylov method on P processors after K iterations and

$$E[T'] = E[\max_p \sum_k \mathcal{T}_p^k] \to K\mu$$

for a pipelined Krylov method, where $\mathcal{T}_p^k$ follows a distribution with pdf $f(x)$, cdf $F(x)$, and mean μ.

Experimental setup

We run PETSc tutorials using GMRES and PGMRES and collect iteration times $\mathcal{T}_p^k$ by making calls to `MPI_Wtime()` at the beginning and end of each Krylov cycle.

```
for k = 0, 1, ...
   start = MPI_Wtime();

   /* GMRES iteration */

   end = MPI_Wtime() - start;
```

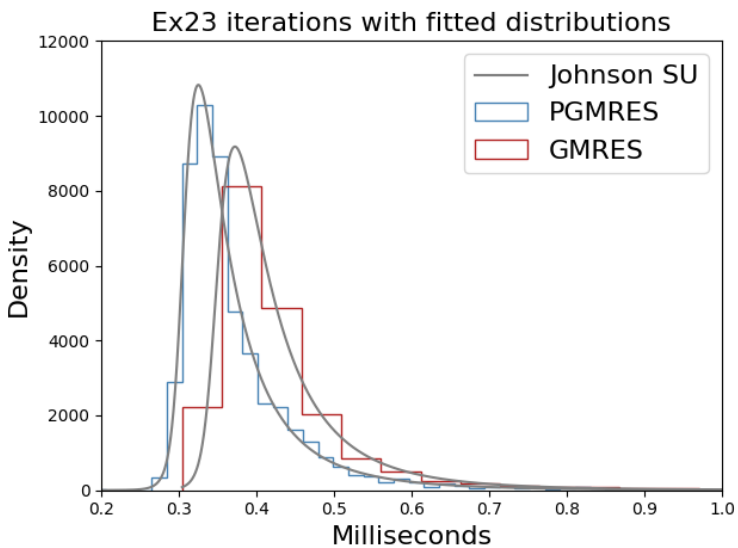
Argonne NATIONAL LABORATORY

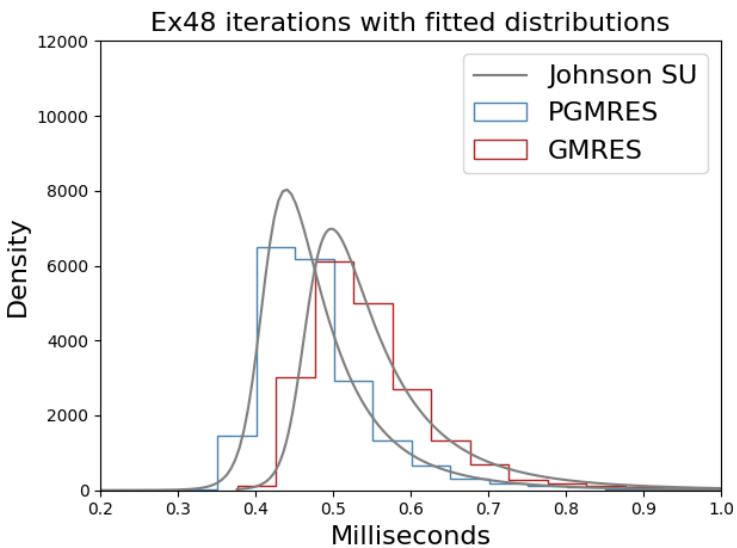
Experimental setup

PETSc KSP tutorial ex23 is a one-dimensional discretization of the Laplacian and SNES tutorial ex48 solves the hydrostatic equations for ice sheet flow. We choose a problem with 10^6 unknowns, use a Jacobi preconditioner, and force 5000 iterates of the Krylov method. We stop after one nonlinear iteration of ex48 and utilize the Cray XC40 Theta at the Argonne Leadership Computing Facility for these runs.

Distribution fitting

We use Python's `scipy` package and calculate the sum of squared error to fit distributions to our data $\mathcal{T}_p^k$ for all p and k. A number of distributions including Johnson SU and non-central Student's T were reasonable fits.





Argonne NATIONAL LABORATORY

Results

We calculate $E[T]$ and $E[T']$ and compare our estimates to the KSPSolve time provided by the PETSc `-log_view` file, shown below for PETSc tutorials ex23 and ex48. Times are in seconds.

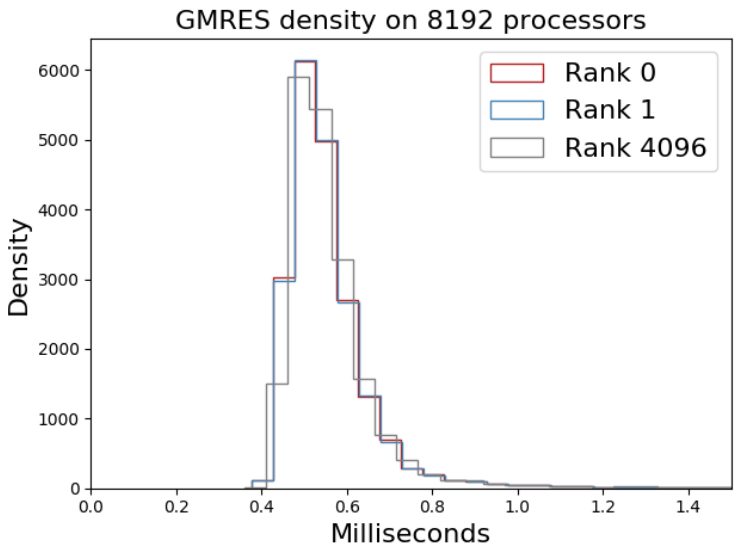
	ex23		ex48	
	GMRES	PGMRES	GMRES	PGMRES
KSPSolve	2.217	2.006	2.943	2.656
$E_{\text{JohnSU}}[T]$	4.831	1.865	5.716	2.413

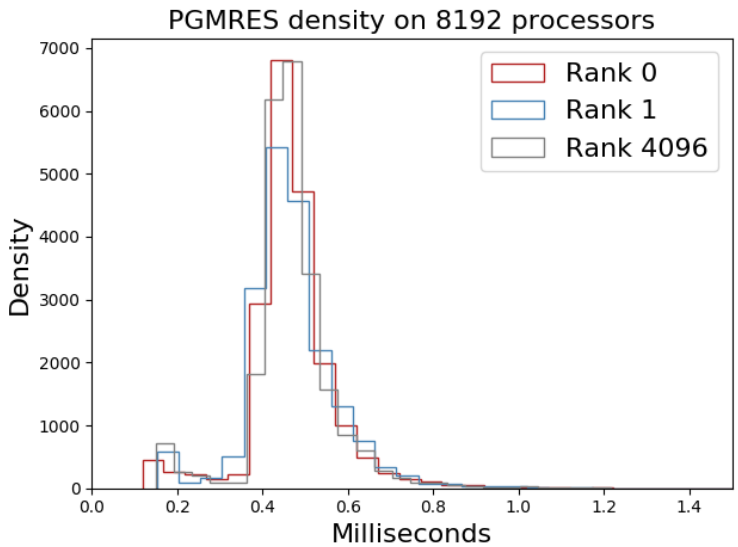
The pipelined model is a reasonable estimate, but this Krylov model is not a good one.

Argonne NATIONAL LABORATORY

Experimental results

A histogram of the iterates $\mathcal{T}_p^k$ on fixed ranks and use of the two sample Kolmogorov-Smirnov test show that GMRES iterates are identical with respect to process, but not PGMRES.

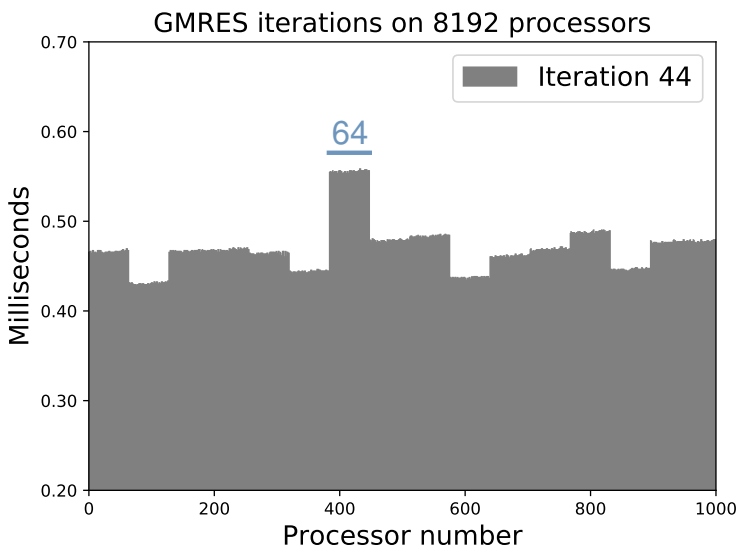


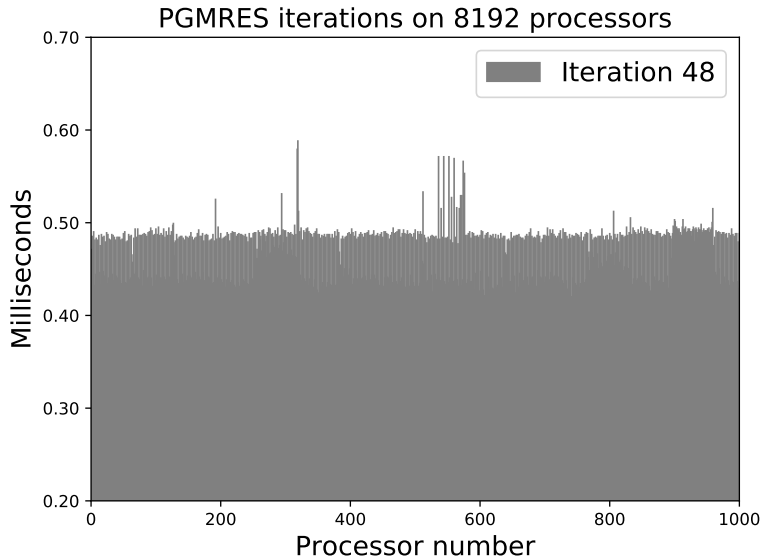


Argonne NATIONAL LABORATORY

Experimental results

A bar graph shows that GMRES iteration times are nearly constant on each KNL node (so they are not independent with p), but there is not much of a difference between PGMRES nodes.

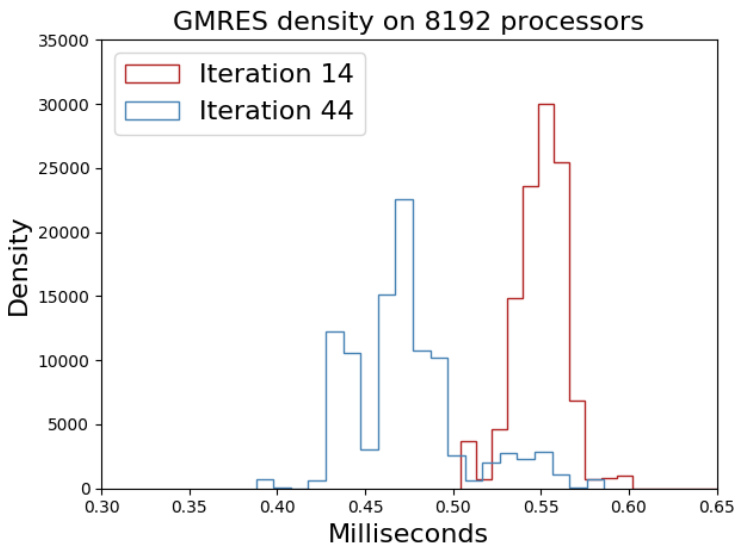


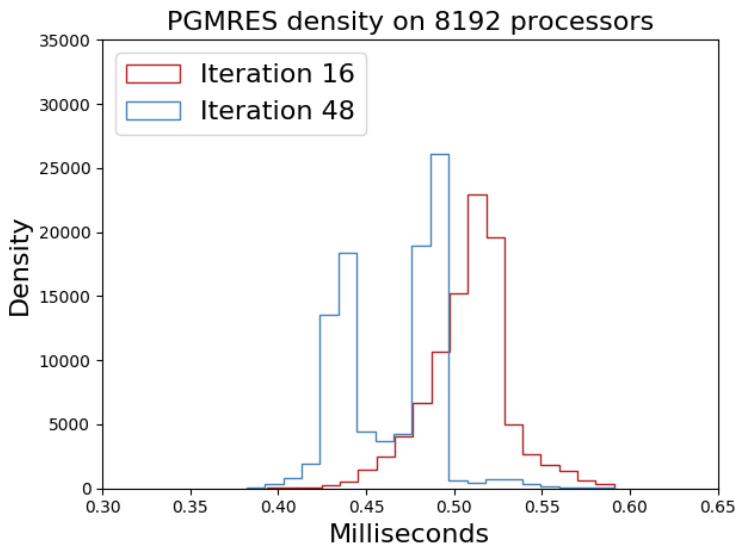


Argonne NATIONAL LABORATORY

Experimental results

A histogram of times for fixed iterations shows that the random variables $\mathcal{T}_p^k$ are not stationary in time (k). Furthermore, they do not resemble any well-known family of distributions.

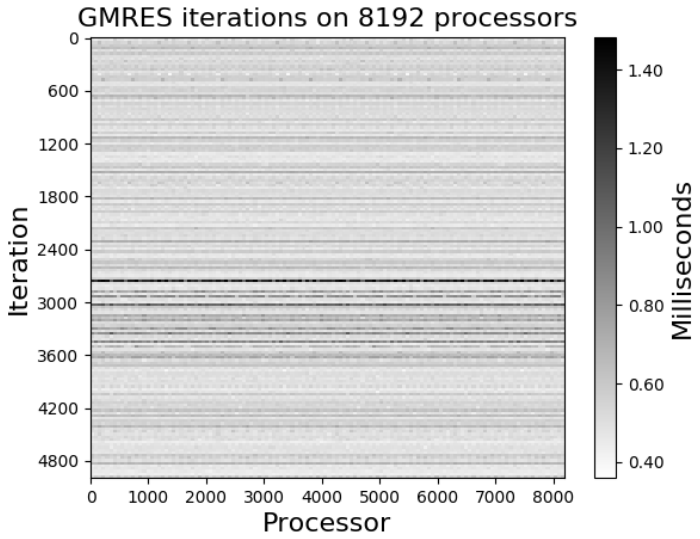


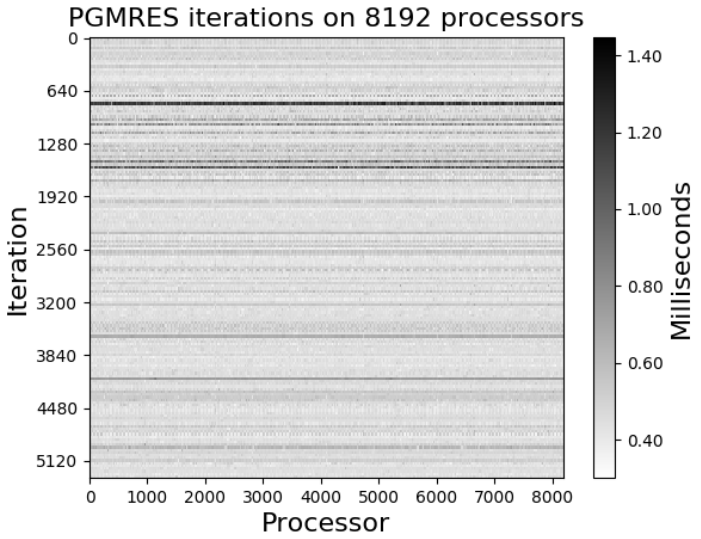


Argonne NATIONAL LABORATORY

Experimental results

A 2-color colormap shows again that the iterates are not stationary in time, but are very similar within an iteration. Clusters of longer iterations could be explained by operating system interruptions.

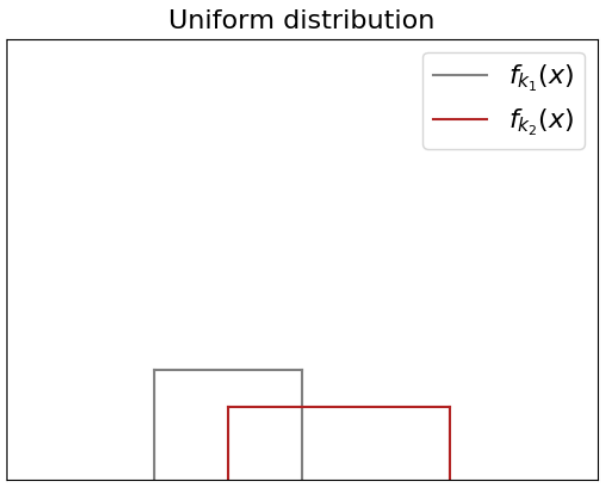




Argonne NATIONAL LABORATORY

Non-stationary performance model

Based on these results, we turn to a non-stationary performance model where the distribution of $\mathcal{T}_p^k$ is allowed to fluctuate across iterations.



For example, we might have $\mathcal{T}_p^{k_1} \sim \text{Uniform}(1, 1)$ but $\mathcal{T}_p^{k_2} \sim \text{Uniform}(1.5, 1.5)$.

Argonne NATIONAL LABORATORY

Non-stationary performance model

Assuming that runtimes are identical and independent of process p and allowing them to fluctuate across step k, we get

$$E[T] = \sum_k E[\max_p \mathcal{T}_p^k] = \sum_k P \int_{-\infty}^{\infty} x F_k(x)^{P-1} f_k(x) dx$$

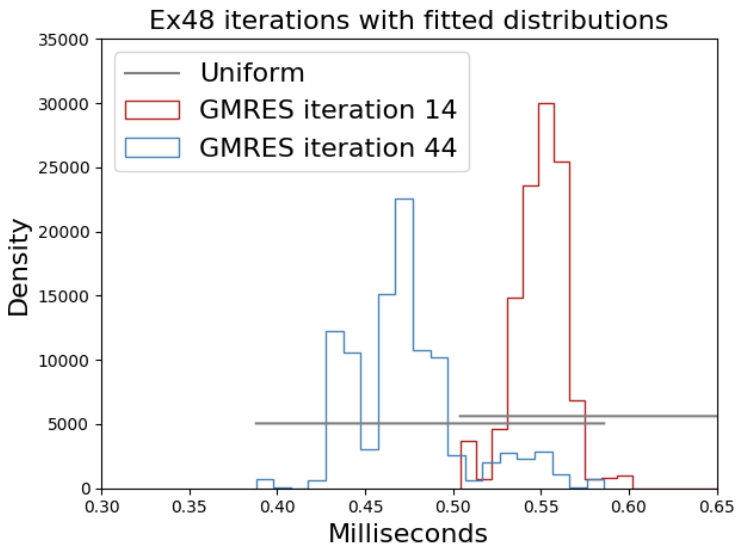
for a Krylov method on P processors after K iterations and

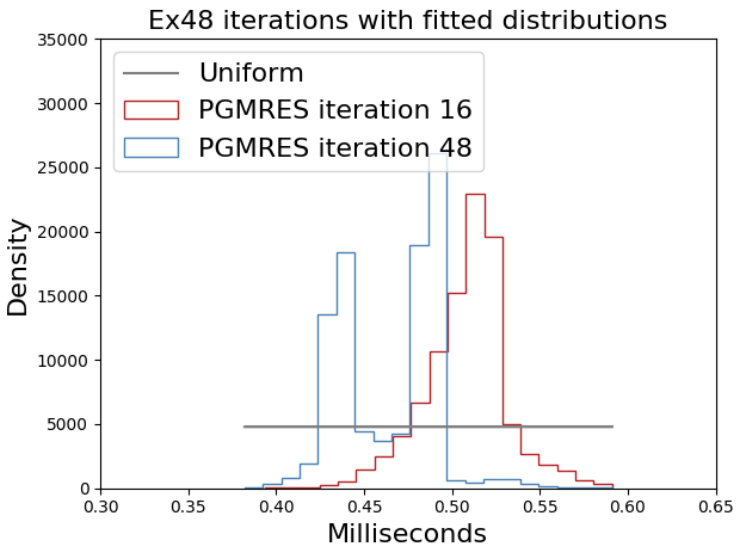
$$E[T'] = E[\max_p \sum_k \mathcal{T}_p^k] \approx \sum_k \mu_k$$

for a pipelined Krylov method, where $\mathcal{T}_p^k$ follows a distribution with pdf $f_k(x)$, cdf $F_k(x)$, and mean μ_k in iteration k.

Stochastic performance model

We assume that $\mathcal{T}_p^k \sim \text{Uniform}(a_k, s_k)$ for all k and fit each iteration to a uniform distribution to find $f_k(x)$, $F_k(x)$, and μ_k.





Argonne NATIONAL LABORATORY

Results

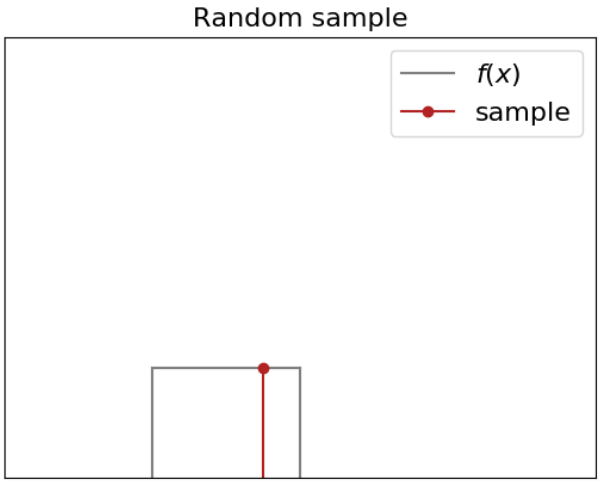
We use our non-stationary stochastic models to calculate $E[T]$ and $E[T']$ and compare our estimates to the KSPSolve time, shown below for PETSc tutorials ex23 and ex48.

	ex23		ex48	
	GMRES	PGMRES	GMRES	PGMRES
KSPSolve	2.217	2.006	2.943	2.656
$E_{\text{Unif}}[T]$	2.432	1.857	3.189	2.455

Using collected iteration data, our performance models are in close agreement with reality.

Predicting runtimes

We perform a random sampling process to predict Krylov and pipelined Kryov runtimes.



If we had a function of a random variable $g(X)$ where $X \sim f(x)$, we could approximate the expected value of g by sampling f.

Argonne NATIONAL LABORATORY

Predicting runtimes

The quantities we want to compute $E[T]$ and $E[T']$ depend on a set of distributions $f_k(x)$, which we will assume are uniform.

The pdf and cdf of a uniform distribution are given by

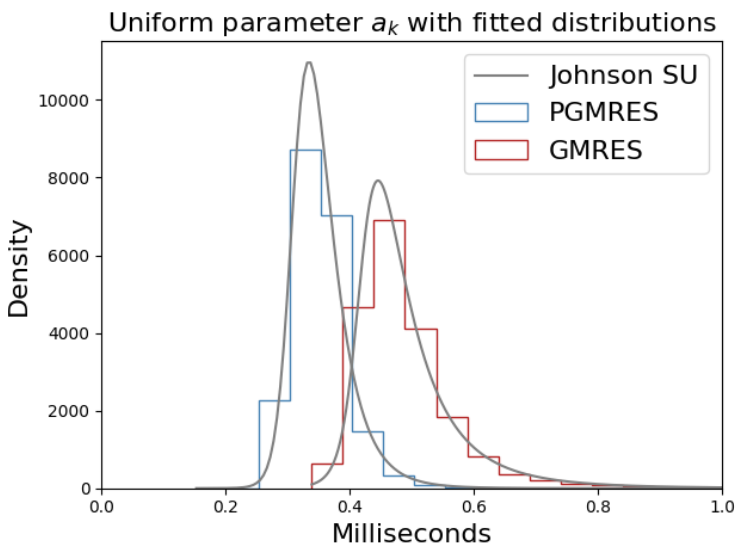
$$f_k(x) = \frac{1}{b_k - a_k}, \quad F_k(x) = \frac{x - a_k}{b_k - a_k}$$

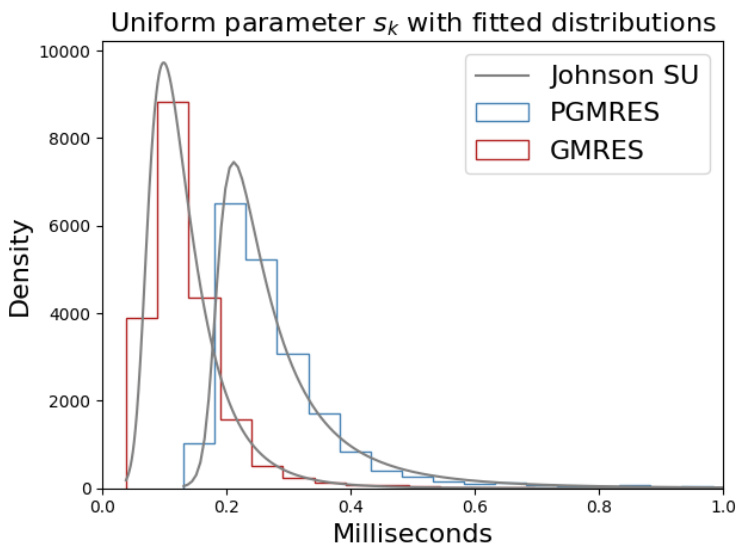
so that it can be described with two parameters: the minimum a_k and span $s_k = b_k - a_k$. These are referred to as `loc` and `scale` in Python.

Then a Krylov method is modeled by K uniform distributions with two sets of descriptive parameters $\{a_k\}$ and $\{s_k\}$.

Predicting runtimes

We model the uniform parameters a_k and s_k as random variables themselves and use `scipy` again to find a good fitting distribution.





Argonne NATIONAL LABORATORY

Predicting runtimes

We simulate a Krylov method by sampling the Johnson SU distributions for a_k and s_k using Scipy's `rvs` and using our performance models to calculate the expected time for that iteration, repeating K times. Results are shown below.

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

When we know the distribution of a_k and s_k, we can reasonably predict $E[T]$ and $E[T']$.

Argonne NATIONAL LABORATORY

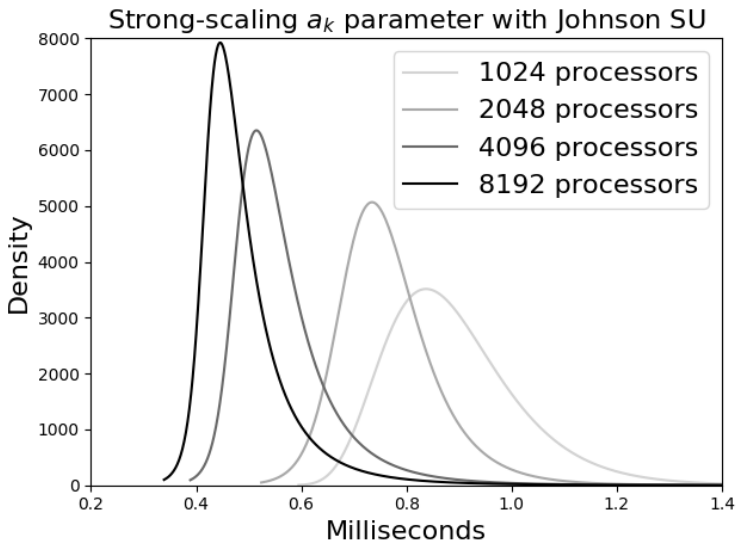
Experimental results

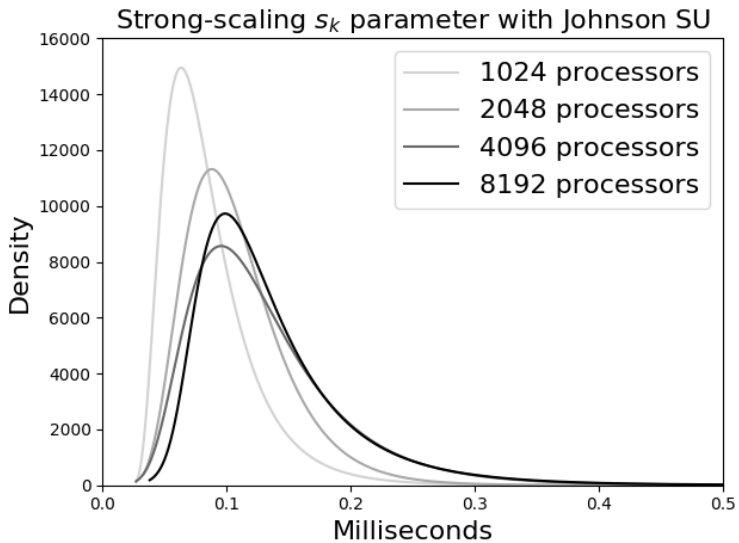
So far, we have shown results from runs of PETSc KSP ex23 and SNES ex48 using GMRES and PGMRES.

Many factors, such as problem size and computing platform will influence the performance of a simulation. By performing a variety of experiments, we see how some of these affect the uniform parameters $\{a_k\}$ and $\{s_k\}$ we use to describe our performance model.

Experimental results

We perform strong-scaling experiments on Theta by changing processor count for a fixed problem size 10^6 unknowns. By decreasing processor count, minimum iteration time grows, but iterations have a shorter span.

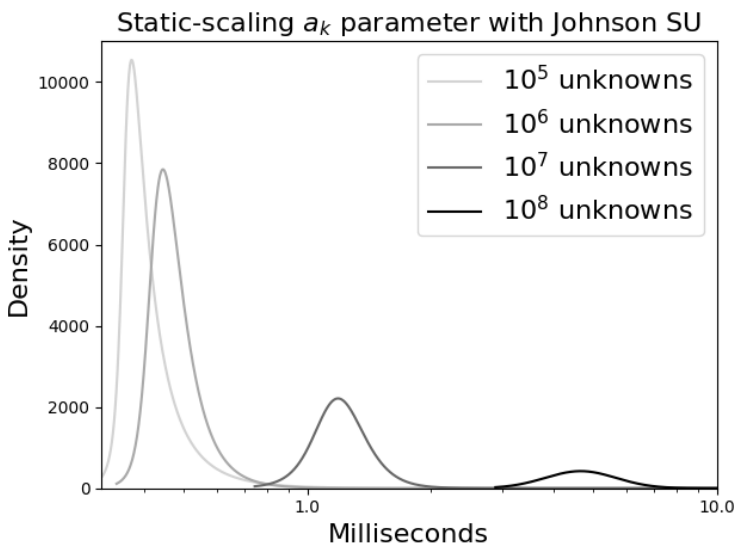


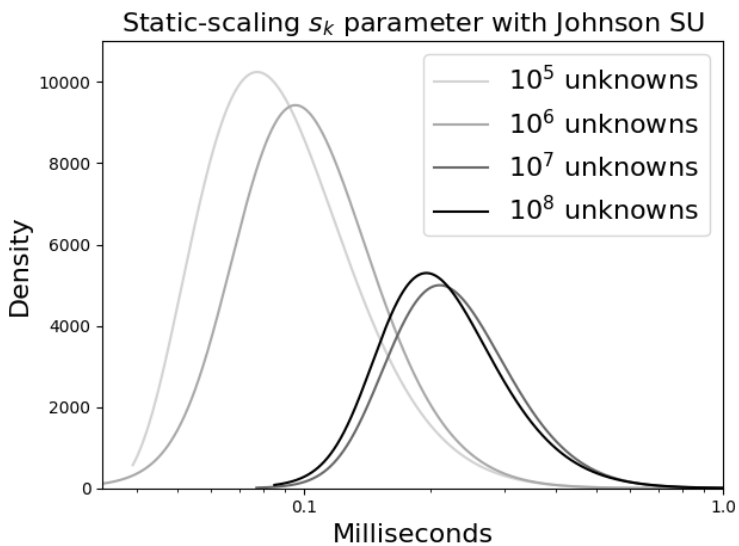


Argonne NATIONAL LABORATORY

Experimental results

We perform static-scaling experiments on Theta by fixing $P = 8192$ and changing the number of unknowns. With more unknowns, iterations take longer and have larger spans. Similar results were found for PGMRES.

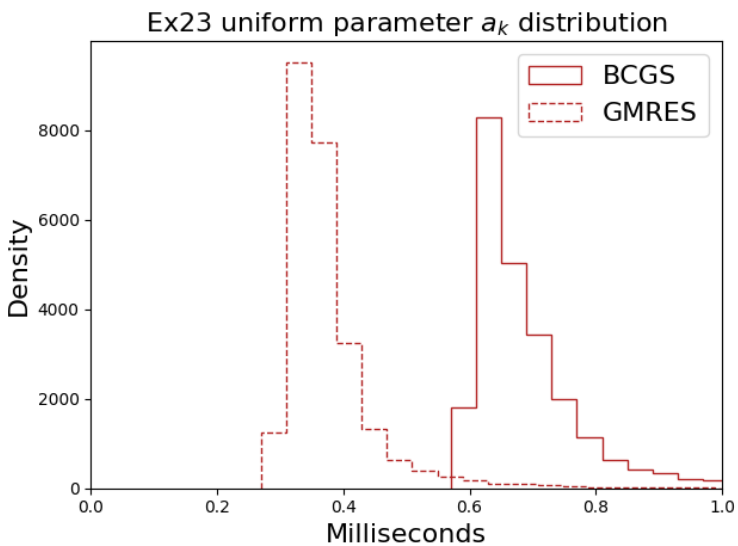


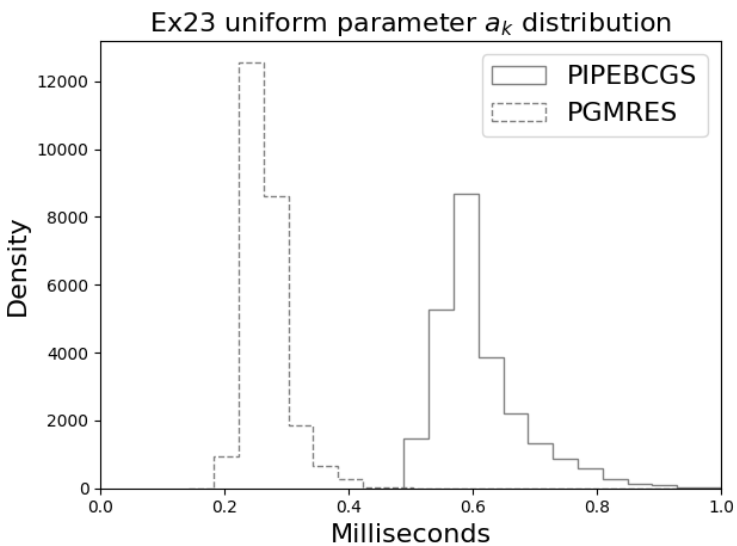


Argonne NATIONAL LABORATORY

Experimental results

We solve PETSc tutorial ex23 using the Stabilized biconjugate gradient method (BCGS) and a pipelined version (PIPEBCGS) on Theta. GMRES and PGMRES have quicker minimum iteration times (uniform parameter a_k) than BCGS and PIPEBCGS.

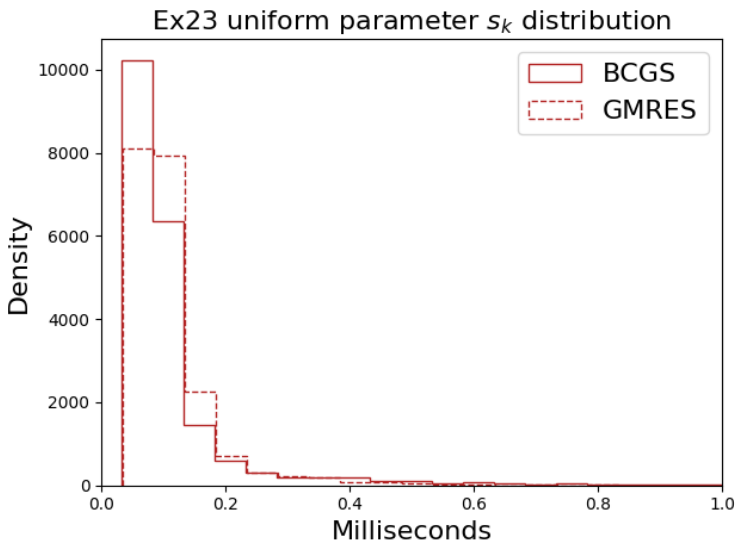


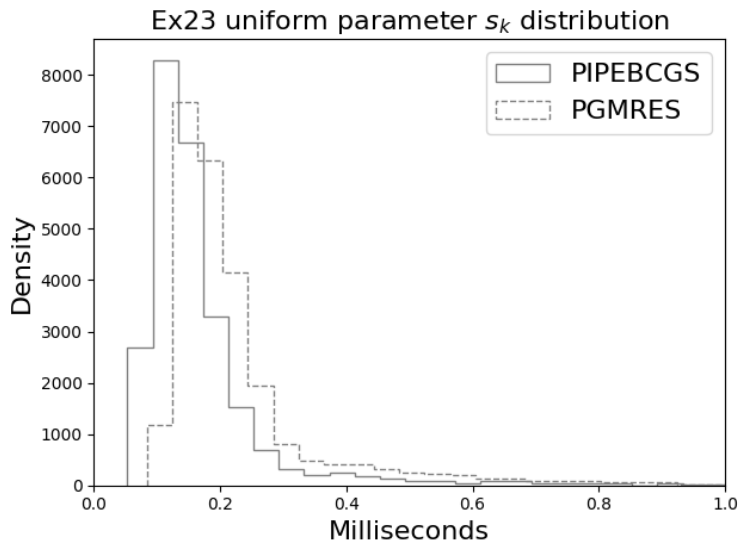


Argonne NATIONAL LABORATORY

Experimental results

The span of iteration times (uniform parameter s_k) are very similar for GMRES and BCGS. A good model needs to account for different performance between Krylov methods.

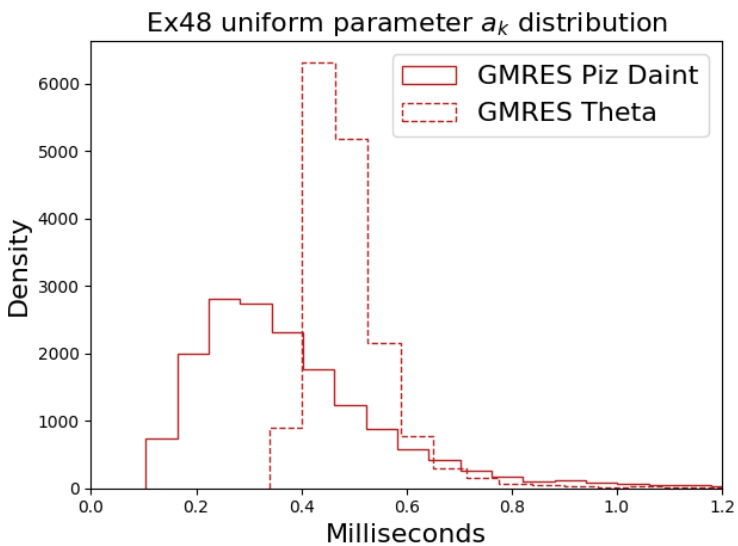


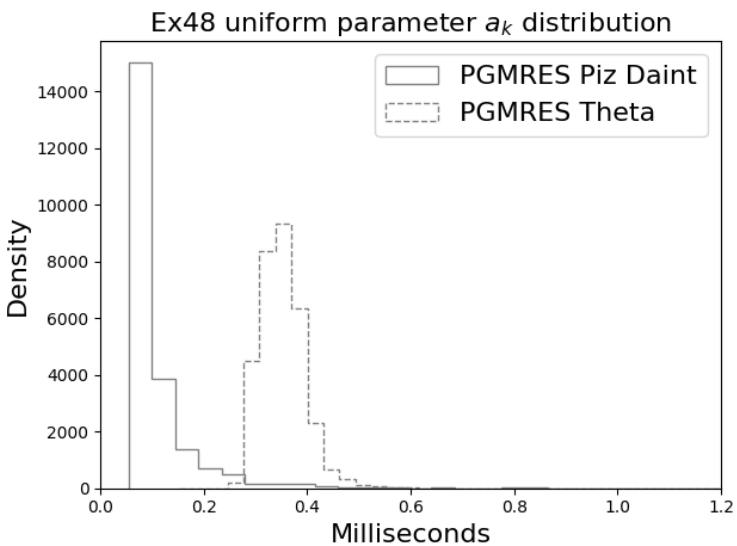


Argonne NATIONAL LABORATORY

Experimental results

We repeat experiments on the Cray XC40 Piz Daint at the Swiss National Computing Center, which contains Intel Xeon E5 Haswell processors on compute nodes. In a given iteration, the quickest rank (uniform parameter a_k) can be much faster on Piz Daint than Theta.

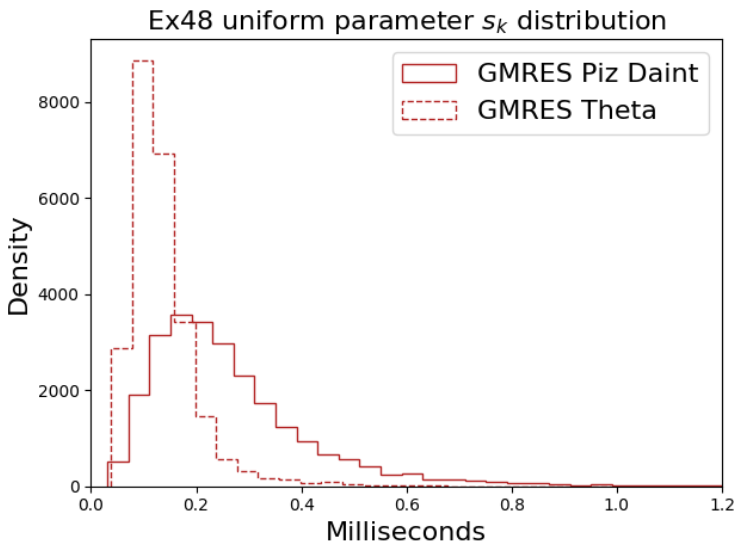


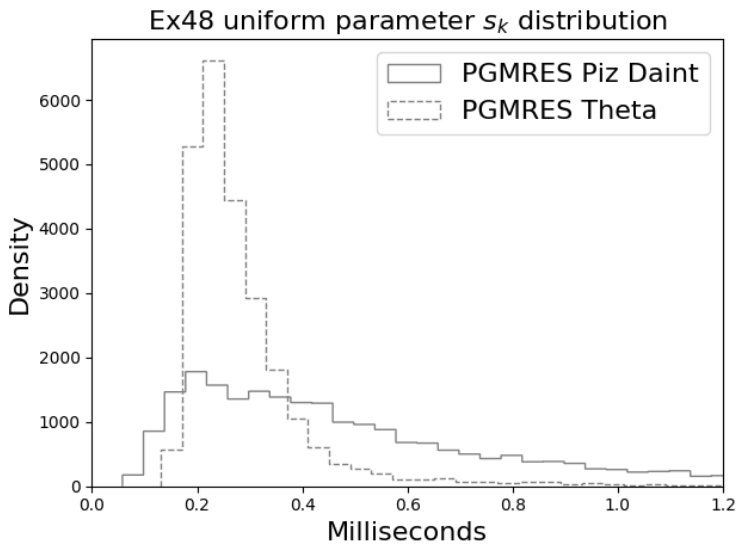


Argonne NATIONAL LABORATORY

Experimental results

The length of an iteration on Piz Daint (uniform parameter s_k) contains much more variation than on Theta , particularly for PGMRES. A robust performance model needs to be flexible enough to account for hardware differences between machines.

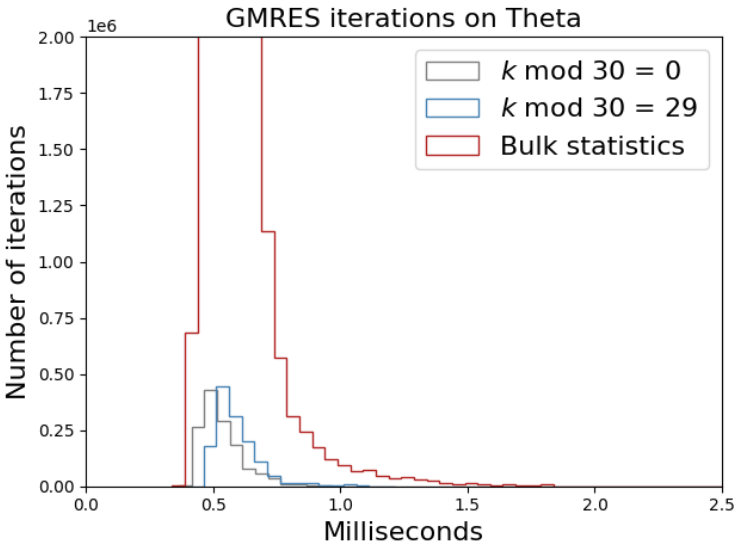


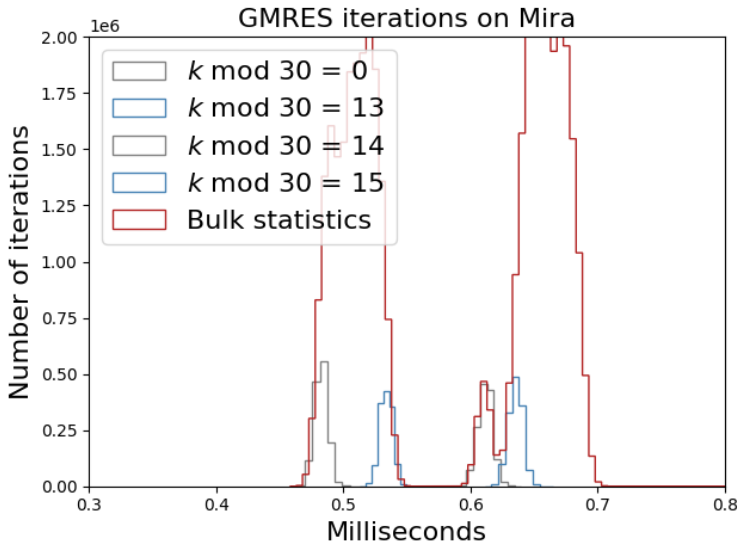


Argonne NATIONAL LABORATORY

Experimental results

We repeat ex48 runs on Mira, an IBM Blue Gene/Q at Argonne's ALCF. Nodes on Mira are connected by a 5D Torus Network with hardware to assist collective functions. A refined model that accounts for varying computation with iteration is needed for good performance estimates on a dedicated network.





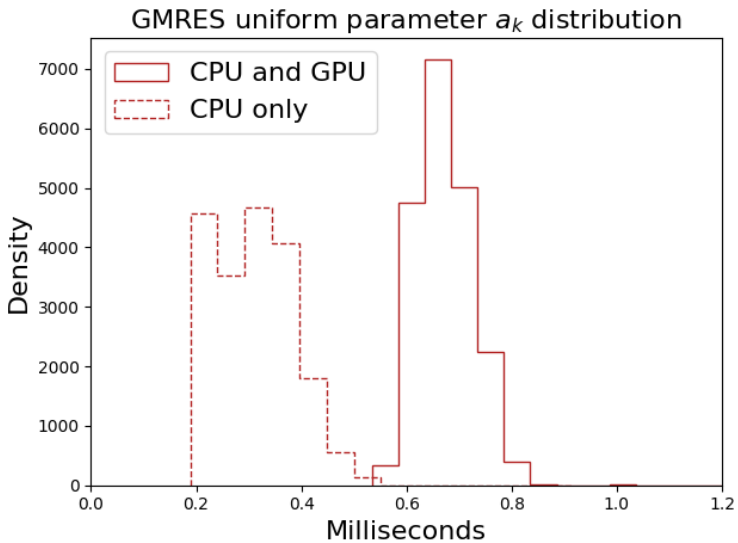
Argonne NATIONAL LABORATORY

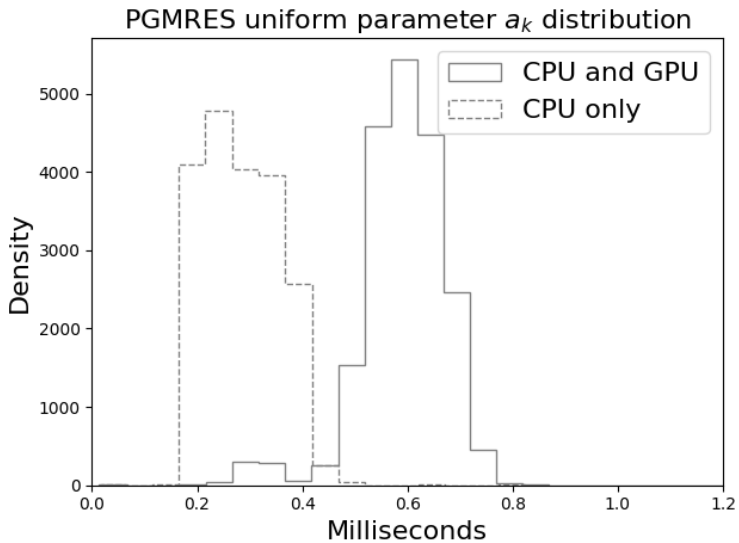
Experimental results

We repeat runs of SNES ex48 with 10^6 unknowns on 8 nodes of Summit at OLCF. On each node of Summit, we use all 42 CPUs with one MPI rank per CPU and compare CPU only experiments to those that utilize the 6 GPUs on each node.

Experimental results

First we look at the uniform parameter a_k, which represents the fastest processor in each iteration. We see that for both GMRES and PGMRES, invoking the GPUs shifts the distribution.

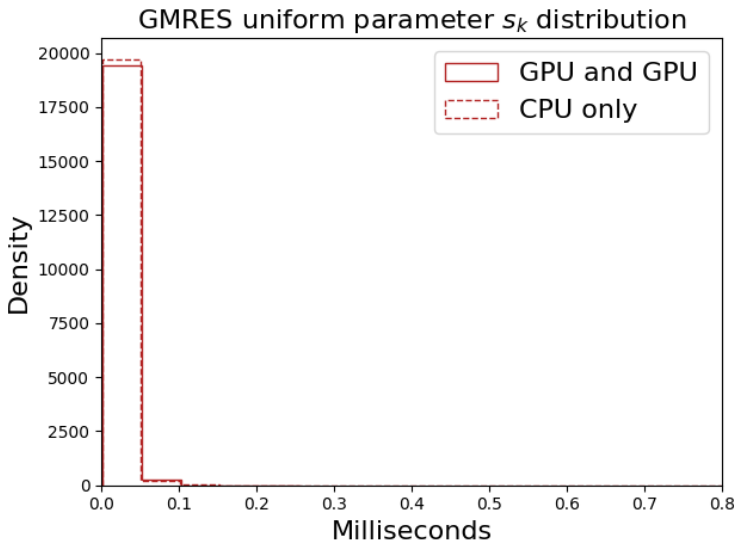


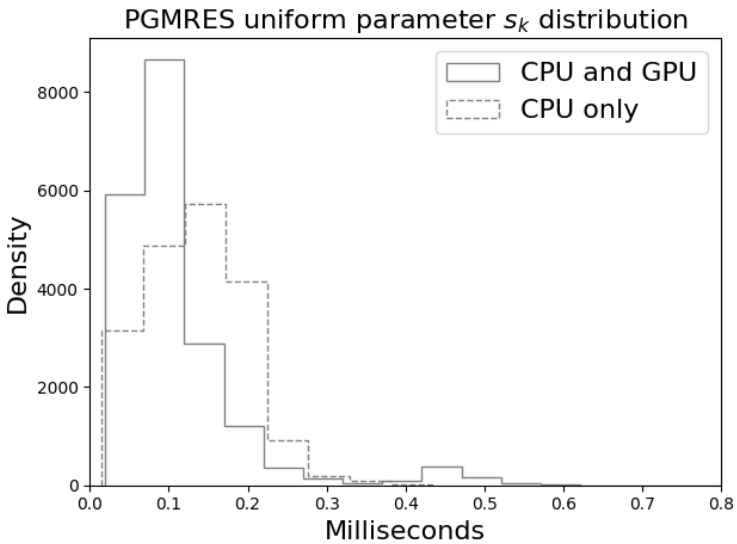


Argonne NATIONAL LABORATORY

Experimental results

The span of times in any given iteration (s_k) are very small for GMRES on Summit, in both GPU and GPU and CPU cases.

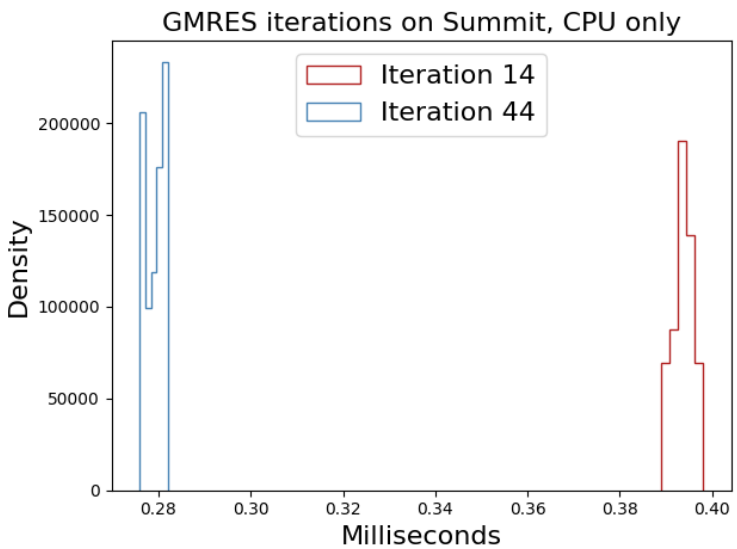


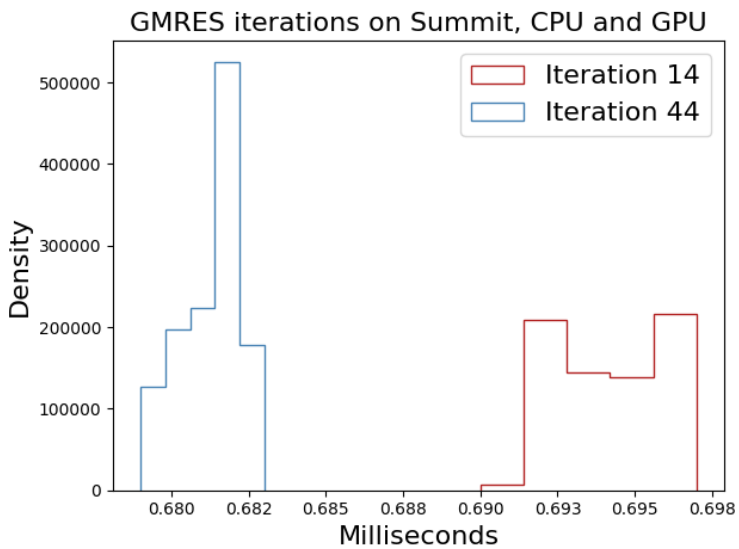


Argonne NATIONAL LABORATORY

Experimental results

The previous slide suggested that GMRES iterations truly perform in lockstep on Summit. Here, we can see that within an iteration, processors execute in nearly the same amount of time whether GPUs are utilized or not.

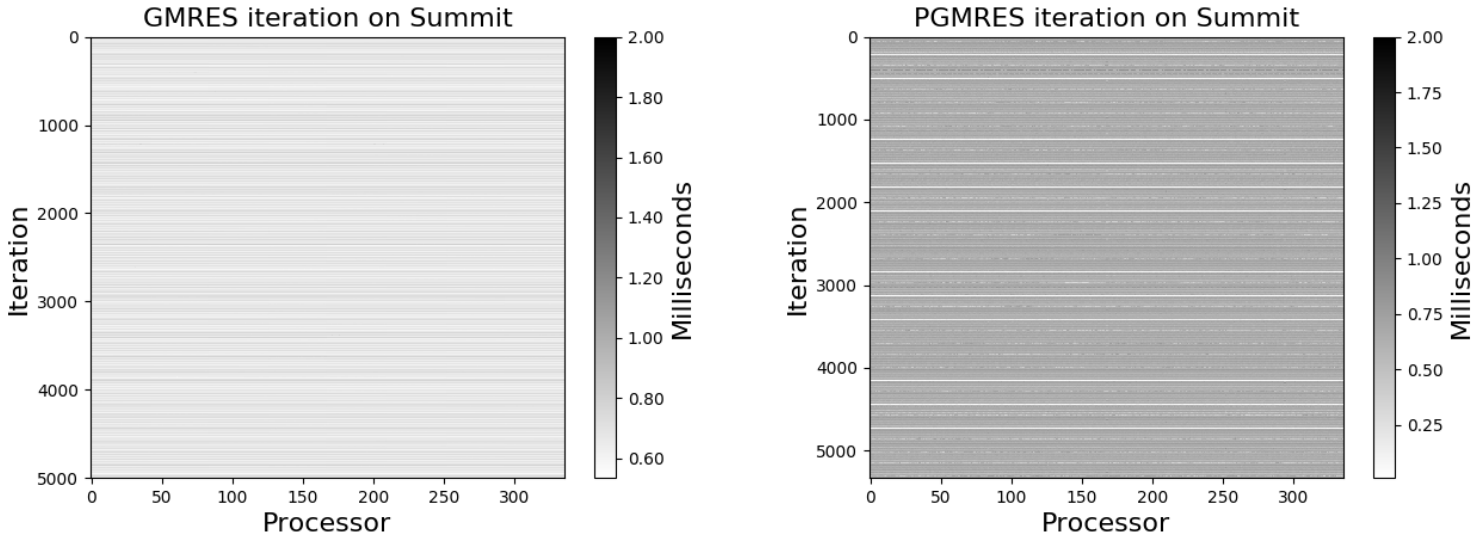




Argonne NATIONAL LABORATORY

Experimental results

The horizontal lines visible in a 2-color colormap shows again that times within an iteration are clustered.



Preliminary results show that a non-stationary performance model could be useful in modeling performance on GPUs.

Argonne NATIONAL LABORATORY

Conclusion

By collecting fine-grained iteration data from runs of Krylov and pipelined Krylov methods, we developed and tested performance models that are in good agreement with reality.

This work shows that we should approach performance modeling in a statistical sense, particularly in high latency situations where we expect unpredictable delays, such as heavily loaded machines or loosely coupled networks such as those used for cloud computing.

This analysis could also perhaps guide the development of new algorithms, particularly those that we expect to be running in less predictable computing environments.

Questions?

Big thanks to Todd Munson, Barry Smith, Ivana Marincic, Vivak Patel, Karl Rupp, and Oana Marin.