Optimal Size of the Block in Block GMRES on GPUs: Computational Model and Experiments

Erik G. Boman [1] *Andrew J. Higgins* [2] Daniel B. Szyld [2]

[1]Center for Computing Research, Sandia National Laboratories

[2]Department of Mathematics, Temple University

April 5, 2022

boratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LL
Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE

$s \times$ GMRES

GMRES is a specific KSM for non-Hermitian systems designed s.t.

$$x_m = \underset{x \in \mathcal{K}_m(A,b)}{\arg\min} \|b - Ax\|_2$$

Say we want to solve a general linear system with multiple RHS $AX = B$ for $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times s}$

- We can set up s separate systems $Ax_i = b_i$ for $i = 1, \ldots, s$
- Using GMRES on each system, we get a solution X_m where:

$$(X_m)_i = \underset{x \in \mathcal{K}_m(A,b_i)}{\arg\min} \|b_i - Ax\|_2$$

- Equivalently,

$$X_m = \underset{X \in \mathcal{K}_m(A,b_1) \times \cdots \times \mathcal{K}_m(A,b_s)}{\arg\min} \|B - AX\|_F$$

Block Krylov Subspaces

Say we want to solve a general linear system with multiple RHS $AX = B$ for $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times s}$

Definition

First, define

$$\mathcal{B}_m(A, B) = \mathcal{K}_m(A, b_1) + \cdots + \mathcal{K}_m(A, b_s)$$

The Block Krylov subspace with m blocks of width s is defined as

$$\mathcal{B}_m^{\square}(A, B) = \underbrace{\mathcal{B}_m(A, B) \times \cdots \times \mathcal{B}_m(A, B)}_{s \text{ times}}$$

NOTE: $\mathcal{K}_m(A, b_1) \times \cdots \times \mathcal{K}_m(A, b_s) \subset \mathcal{B}_m^{\square}(A, B)$

Overview of Block GMRES

Block GMRES (BGMRES) is a specific block KSM for non-Hermitian systems designed s.t.

$$X_m = \underset{X \in \mathcal{B}_m^{\square}(A,B)}{\arg\min} \|B - AX\|_F$$

IMPORTANT!

Since $\mathcal{K}_m(A, b_1) \times \cdots \times \mathcal{K}_m(A, b_s) \subset \mathcal{B}_m^{\square}(A, B)$, BGMRES will converge in fewer iterations than GMRES on each of the s RHS separately

Overview of Block GMRES

The following shows the solution space for each column of the computed solution X for $s \times$ GMRES vs BGMRES

	x_1	x_2	$\cdots$	x_s
	$\cap$	$\cap$		$\cap$
$s \times$ GMRES:	$\mathcal{K}_m(A, b_1)$	$\mathcal{K}_m(A, b_2)$		$\mathcal{K}_m(A, b_s)$
BGMRES:	$\mathcal{K}_m(A, b_1) + \cdots + \mathcal{K}_m(A, b_s)$	$\mathcal{K}_m(A, b_1) + \cdots + \mathcal{K}_m(A, b_s)$		$\mathcal{K}_m(A, b_1) + \cdots + \mathcal{K}_m(A, b_s)$

- As s increases, each column of the BGMRES solution uses more information, so the number of iterations to converge decreases.
- This is not the case for $s \times$ GMRES.

Overview of Work: Two Messages

Motivation

- GPUs excel with parallel tasks that block Krylov methods rely on (e.g., BLAS3/`spmv`)
- *Sometimes*, block Krylov methods perform better than single RHS methods on CPUs
- Perhaps block Krylov methods are more attractive on GPUs

Given $AX = B$, we compared the runtime of solving the system using:

1. $s \times$ GMRES on a CPU
2. $s \times$ GMRES on a GPU
3. BGMRES on a CPU
4. BGMRES on a GPU

Overview of Work: Two Messages

Message 1

There are cases where $s \times GMRES$ is faster than BGMRES on the CPU, but the reverse holds on the GPU

Overview of Work: Two Messages

Message 1

There are cases where $s \times$ *GMRES* is faster than BGMRES on the CPU, but the reverse holds on the GPU

Message 2

On GPUs, performance gains of non-restarted BGMRES over $s \times$ GMRES are diminished as the number of RHS s grows.

There is an optimal number of RHS s where non-restarted BGMRES is most effective over $s \times$ GMRES

Comparison of Block vs Non-Blocked GMRES

To solve $AX = B$ with $A \in \mathbb{R}^{n \times n}$, s = number of RHS, m = number of iterations until convergence, nz = number of non-zeros in A:

Algorithm BGMRES: m iter.

1: $B = B - AX_0$
2: $[V_1, \beta] = \text{qr}(B)$
3: **for** $j = 1, \ldots, m$ **do**
4: $W = AV_j$
5: **for** $i = 1, \ldots, j$ **do**
6: $H_{i,j} = W^T V_i$
7: $W = W - H_{i,j} V_i$
8: **end for**
9: $[V_{j+1}, H_{j+1,j}] = \text{qr}(W)$
10: **end for**
11: $Y_m = \arg\min \|HY_m - \beta E_1\|_F$
12: $V = [V_1|\ldots|V_m]$
13: $X_m = X_0 + VY_m$

Algorithm $s \times$ GMRES: m iter.

1: **for** $k = 1, \ldots, s$ **do**
2: $B = b_k - A(x_0)_k$
3: $\beta = \|B\|_2, v_1 = B/\beta$
4: **for** $j = 1, \ldots, m$ **do**
5: $w = Av_j$
6: **for** $i = 1, \ldots, j$ **do**
7: $h_{i,j} = w^T v_i$
8: $w = w - h_{i,j} v_i$
9: **end for**
10: $h_{j+1,j} = \|w\|_2, v_{j+1} = w/h_{j+1,j}$
11: **end for**
12: $y_m = \arg\min \|Hy_m - \beta e_1\|_2$
13: $V = [v_1|\ldots|v_m]$
14: $(x_m)_k = (x_0)_k + Vy_m$
15: **end for**

Comparison of Block vs Non-Blocked GMRES

Key differences:

- $s \times$ GMRES requires s times more accesses of A than BGMRES
- BGMRES uses s times more FLOPs in the orthog. step than $s \times$ GMRES

Observation 1

If $m_{BGMRES} \approx \frac{m_{GMRES}}{s}$, BGMRES will be about as fast as GMRES

Observation 2

Even if $m_{BGMRES} \approx m_{GMRES}$, BGMRES can outperform $s \times$ GMRES if FLOPs are cheap and accessing the coefficient matrix A is expensive

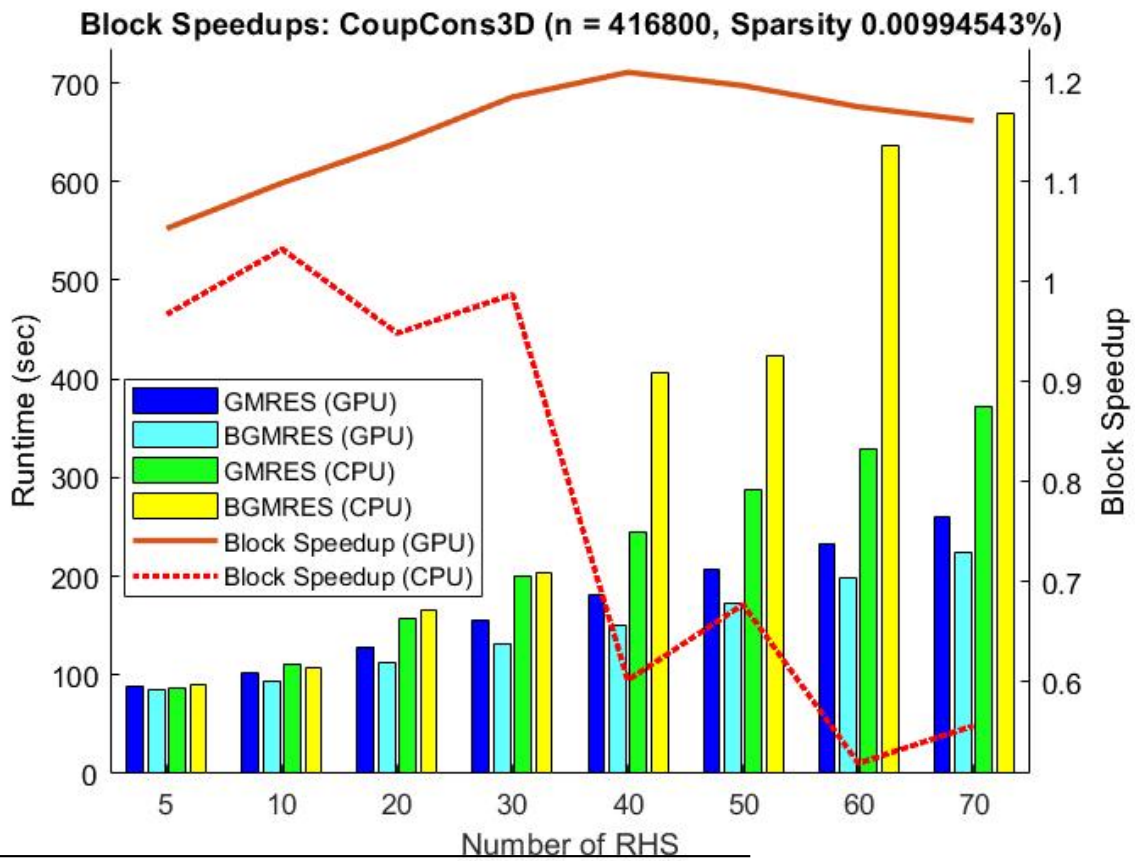
Observation 3

The benefit of BGMRES over $s \times$ GMRES is a balancing act between fewer accesses of A and a more expensive orthog. procedure

Experimental Results: $n = 112,211$

We tested BGMRES/GMRES on matrices from UFSMC on CPUs[1]& GPUs[2]with random RHS, and analyzed

$$\textbf{Block Speedup} = \text{Time}_{GMRES}/\text{Time}_{BGMRES}$$



Message 1

There are cases where $s \times GMRES$ is faster than BGMRES on the CPU, but the reverse holds on the GPU

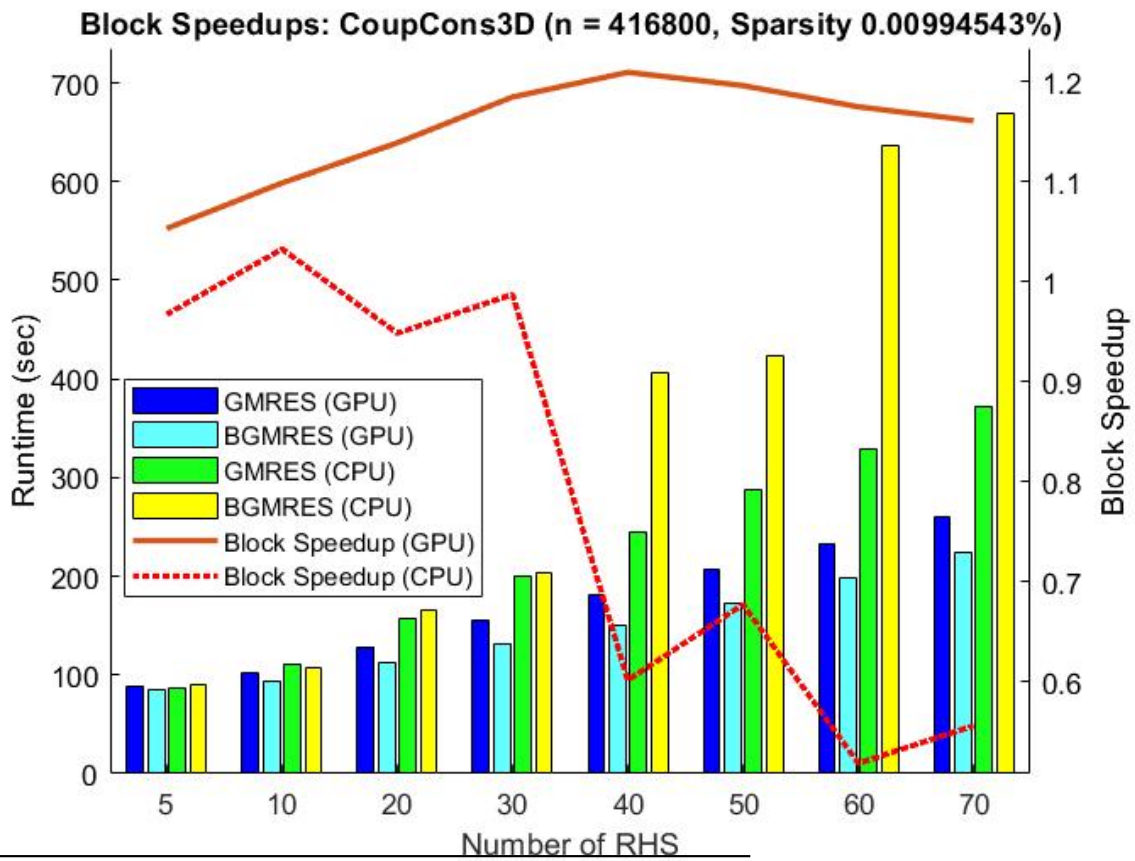
[1] Intel Xeon CPU E5-2698 v4, 20-core, 2.2GHz

[2] NVIDIA Tesla V100 SXM2

Experimental Results: $n = 112,211$

We tested BGMRES/GMRES on matrices from UFSMC on CPUs[1]& GPUs[2]with random RHS, and analyzed

Block Speedup $= \text{Time}_{GMRES}/\text{Time}_{BGMRES}$



[1] Intel Xeon CPU E5-2698 v4, 20-core, 2.2GHz

[2] NVIDIA Tesla V100 SXM2

Message 1

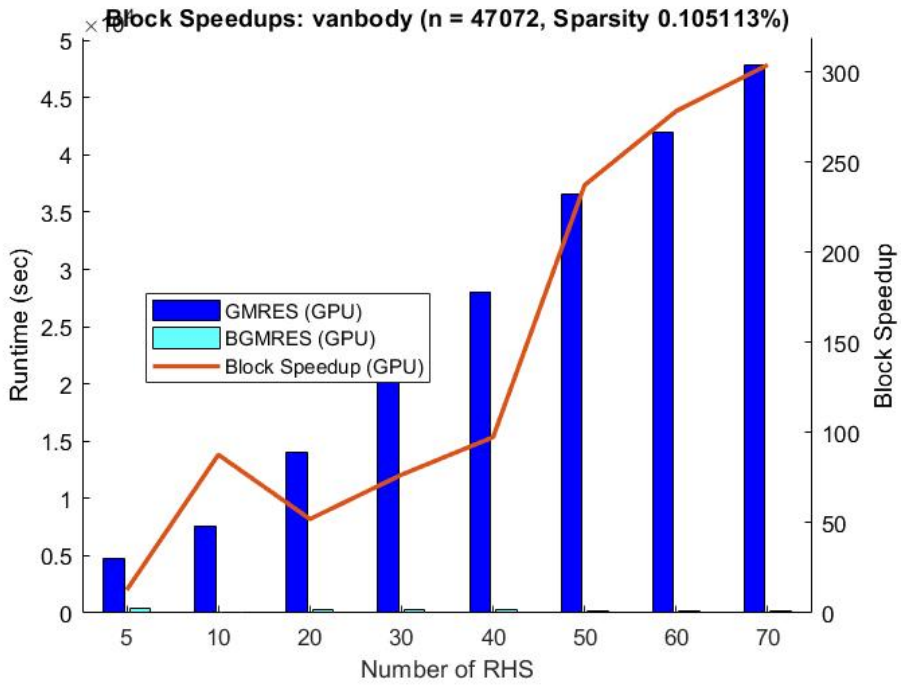
There are cases where $s \times GMRES$ is faster than BGMRES on the CPU, but the reverse holds on the GPU

Message 2

On the GPU, there is an optimal number of RHS s where BGMRES is most effective over $s \times$ GMRES

Experimental Results: $n = 47,072$ (with Restarts)

- Restarts can make convergence behavior less clear
- Do we get a clear optimal number of RHS with restarts?
  - **Not necessarily**



Convergence of BGMRES for `vanbody`		
# of RHS	Restart Parameter	Total Iterations
5	854	493
10	427	324
20	214	198
30	142	239
40	107	291
50	85	413
60	70	215
70	61	243

Theoretical Runtime Model

We built a theoretical model of the runtime for GMRES and BGMRES via a latency-bandwidth model (inspired by Hoemmen).
Split the runtime of the algorithms into two components:

1. **Communication time**
   The time required to send a message of m words is modeled by:

$$\text{Time}_{\text{Message}}(m) = \alpha + \beta \cdot m$$

   $\alpha =$ latency (in sec), $\beta =$ inverse bandwidth (in sec/word)

2. **Floating Point Time**
   Time required to execute m floating point operations is modeled by:

$$\text{Time}_{\text{flops}}(m) = \gamma \cdot m$$

   $\gamma =$ inverse floating point throughput (in sec/flop)

By counting operations, we get runtime models for BGMRES and GMRES

Theoretical Runtime Model

By counting operations, we get runtime models for BGMRES and GMRES:

$$\text{Time}_{\text{BGMRES}}(n, nz, m, s) \approx \overbrace{\alpha \cdot m + \beta \cdot nz \cdot m}^{\text{Data Movement}} + \gamma \cdot (\underbrace{2nz \cdot ms}_{\texttt{spmv}} + \underbrace{2nm^2s^2 + 2nms^2}_{\text{Block Orthogonalization}})$$

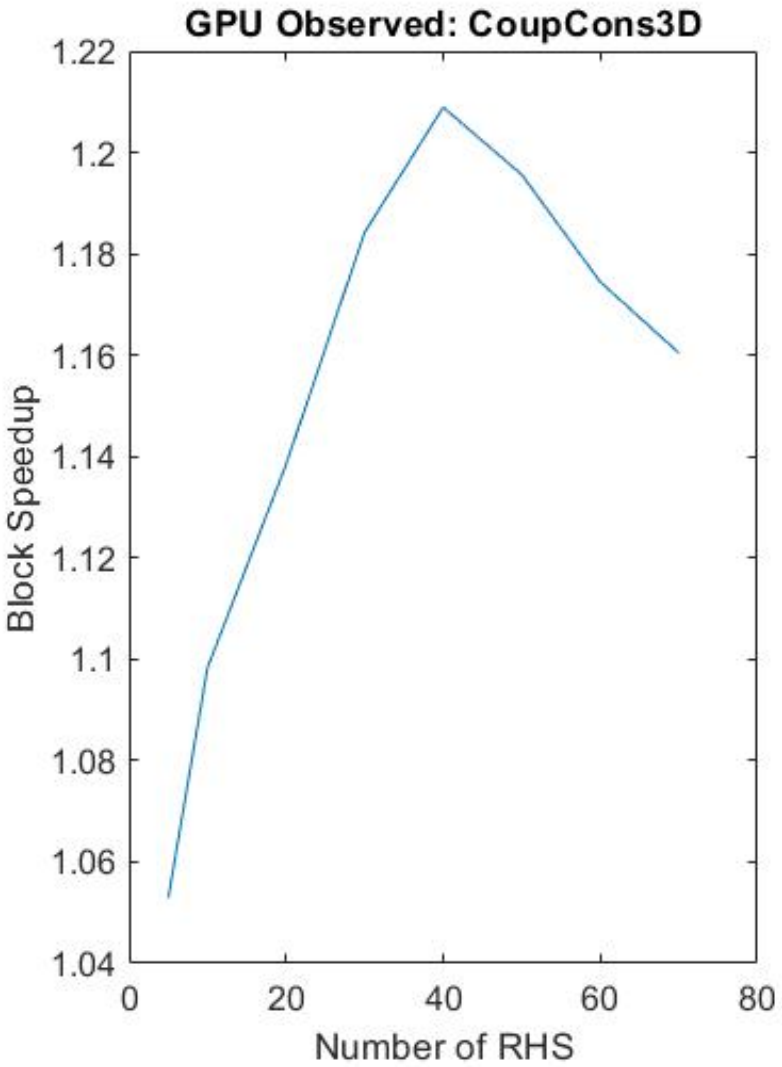
$$\text{Time}_{\text{GMRES}}(n, nz, m, s) \approx \overbrace{\alpha \cdot ms + \beta \cdot nz \cdot ms}^{\text{Data Movement}} + \gamma \cdot (\underbrace{2nz \cdot ms}_{\texttt{spmv}} + \underbrace{2nm^2s + 3nms}_{\text{Orthogonalization}})$$

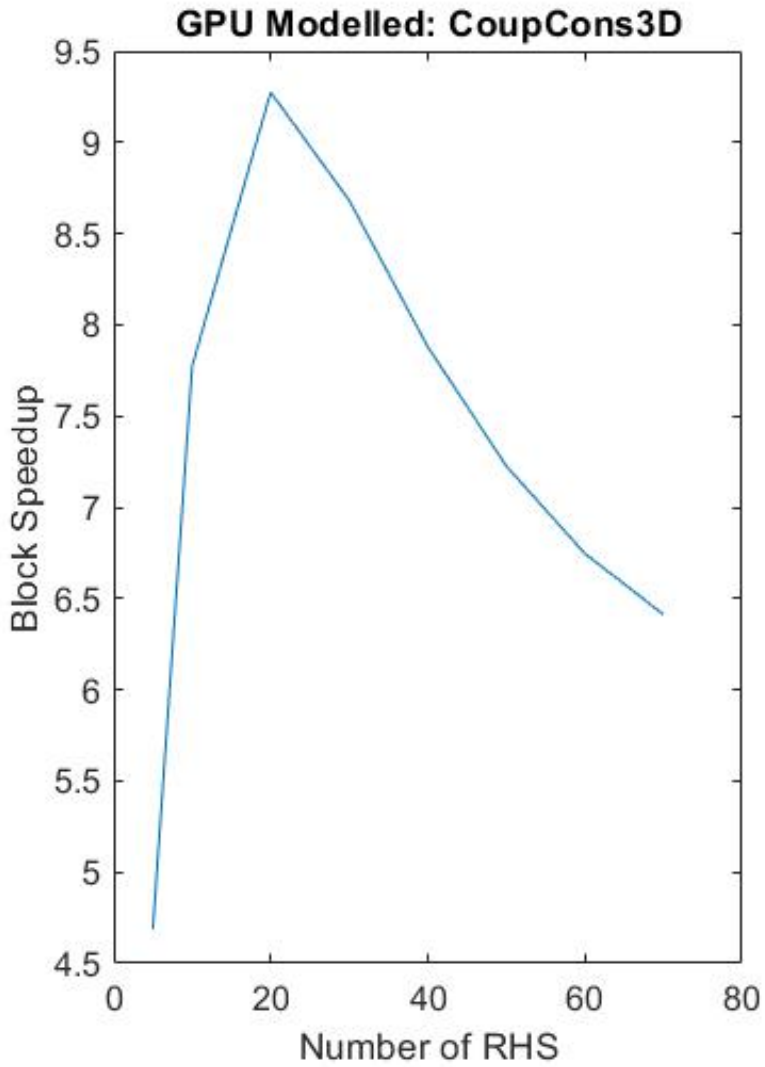
nz = number of non-zeros in $A \in \mathbb{R}^{n \times n}$

s = number of RHS, m = number of iterations required for convergence

Modeled vs Observed Speedups: non-restarted case

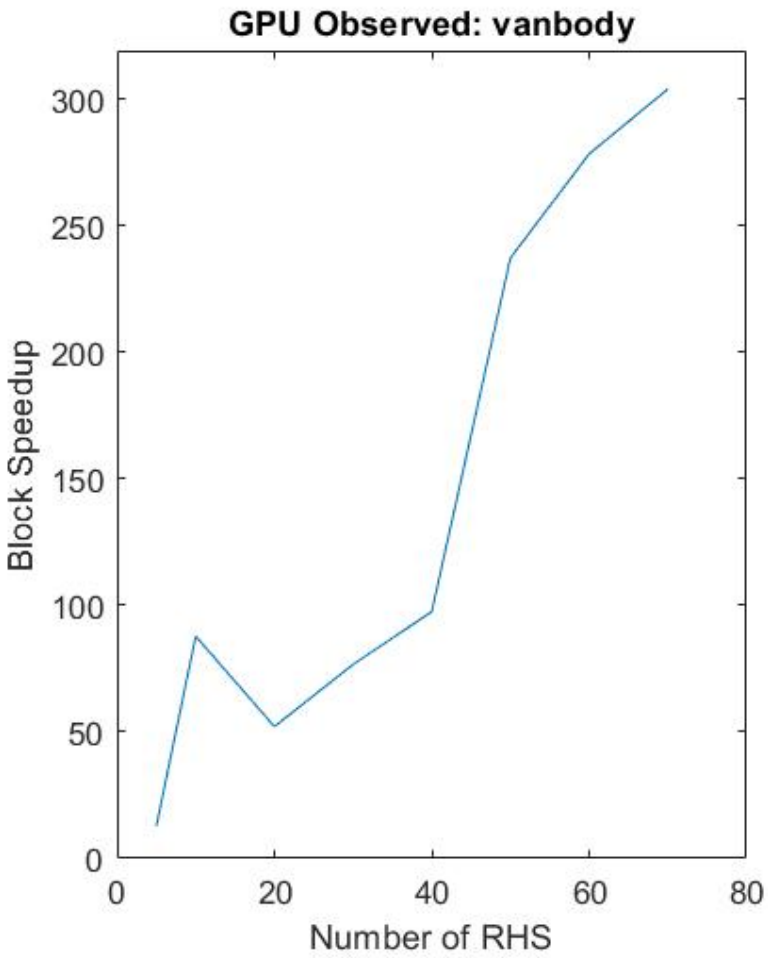
Both observed and modeled Block Speedups have a clear optimal number of RHS

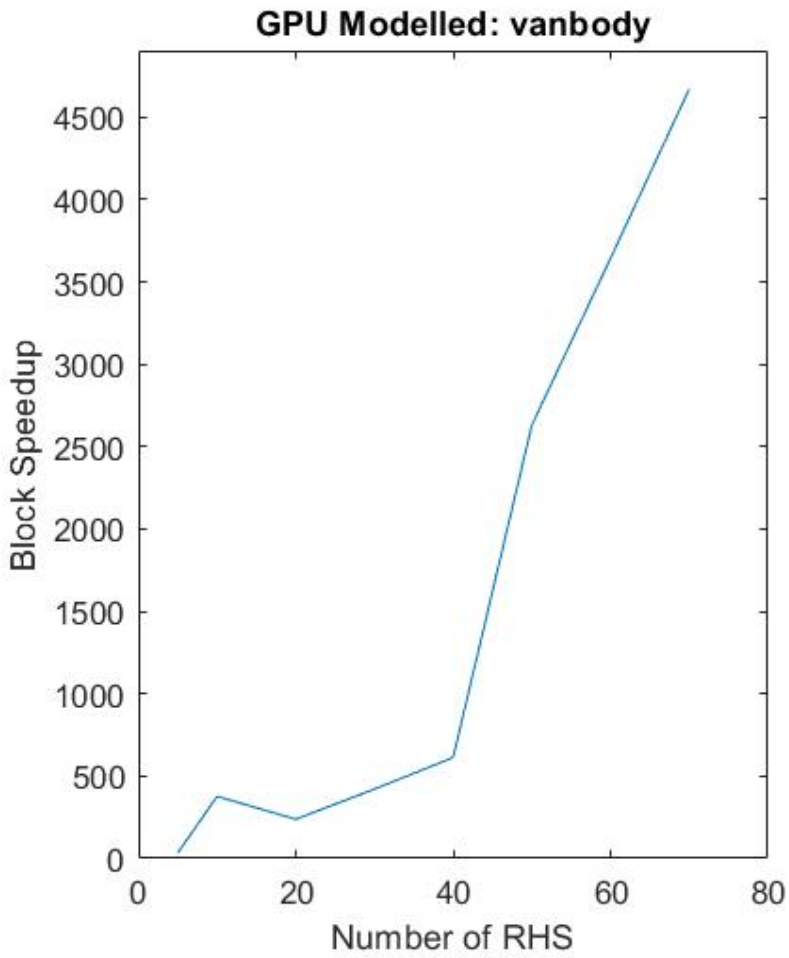




Modeled vs Observed Speedups: restarted case

Although the restarted case exhibits multiple peaks in Block Speedup, the runtime model captures this behavior relatively well





Conclusions: BGMRES vs $s \times$ GMRES

Message 1

There are cases where $s \times$ *GMRES* is faster than BGMRES on the CPU, but the reverse holds on the GPU

Message 2

There is an optimal number of RHS s where non-restarted BGMRES is most effective over $s \times$ GMRES

- **Why?** As s increases, the extra cost of FLOPs in the block orthog. process eventually outweighs the savings in data movement

Our runtime model captures the qualitative behavior of the Block Speedup effectively with and without restarts

Thank You!

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

This work was in part supported by the Exascale Computing Project (17-SC-20- SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

This research includes calculations carried out on HPC resources at Temple University.