



# The ESIF-HPC-2 Benchmark Suite

---

Christopher Chang

Benchmarking in the Datacenter

February 22, 2020

# Acknowledgment

## Co-Leads

- Ilene Carpenter
- Wes Jones

## Design Review Team

## DOE-EERE

## Developers

- Matt Bidwell
- Ilene Carpenter
- Ross Larsen
- Hai Long
- Avi Purkayastha
- Caleb Phillips
- Jon Rood
- Deepthi Vaidhynathan

## Testers

- Shreyas Ananthan
- Ross Larsen
- Hai Long
- Monte Lunacek
- Avi Purkayastha
- Matthew Reynolds
- Jeff Simpson
- Stephen Thomas

# Contents

- 1 Introduction to Datacenter and Context
- 2 Motivations for Creating a Suite
- 3 Contents of Suite, Development, and Configurations
- 4 What do Benchmarks Cover?
- 5 Conclusion

# ESIF-HPC-2 Benchmark Suite

---

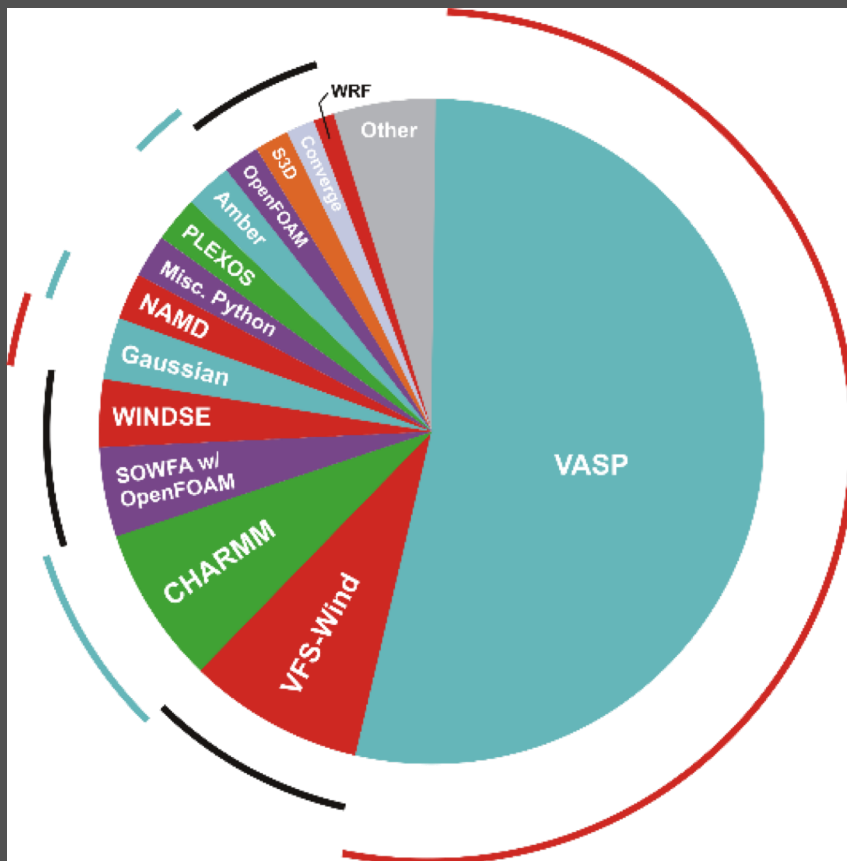
Introduction and Context

# ESIF-HPC at NREL



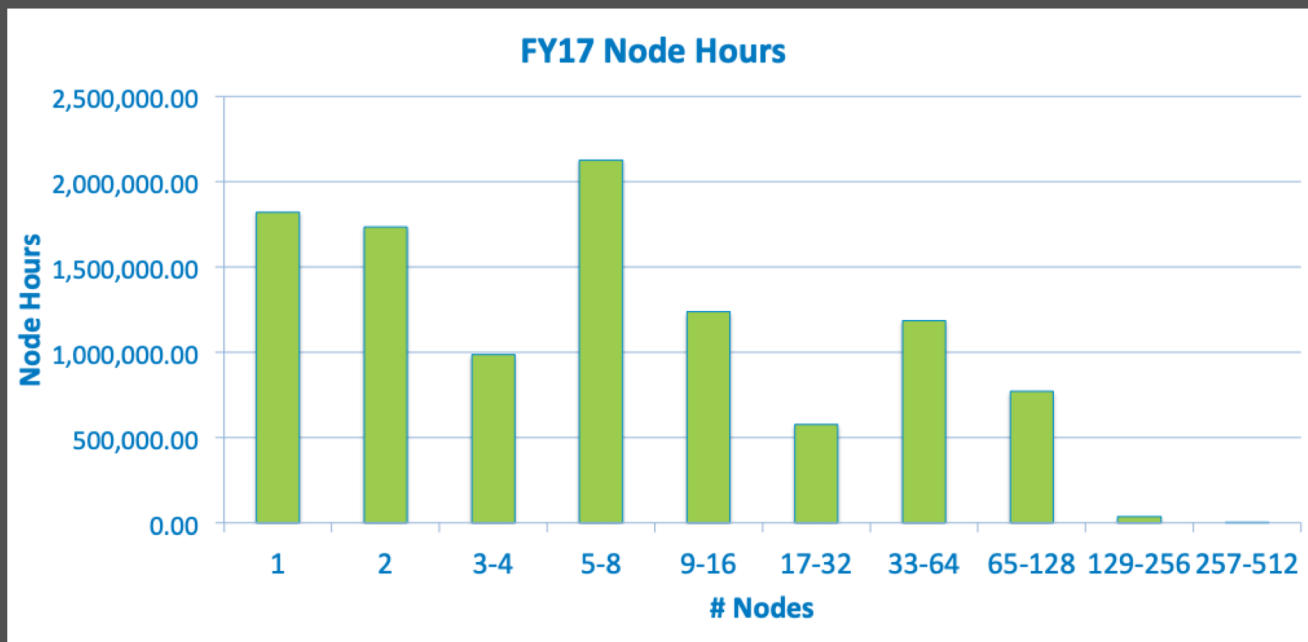
- “the largest HPC environment in the world dedicated to advancing renewable energy and energy efficiency technologies”
- Current production machine is Eagle
  - 8 PF, 2200 2×18-core Intel Skylake nodes
  - 14 PB Lustre PFS
  - 800 TB Qumulo utility NFS
  - 8D hypercube EDR Infiniband

# Peregrine Workload Analysis



- 60% electronic structure
- 20% CFD/multiphysics
- 10% molecular dynamics
- 10% other (Python, workflow, postprocessing)

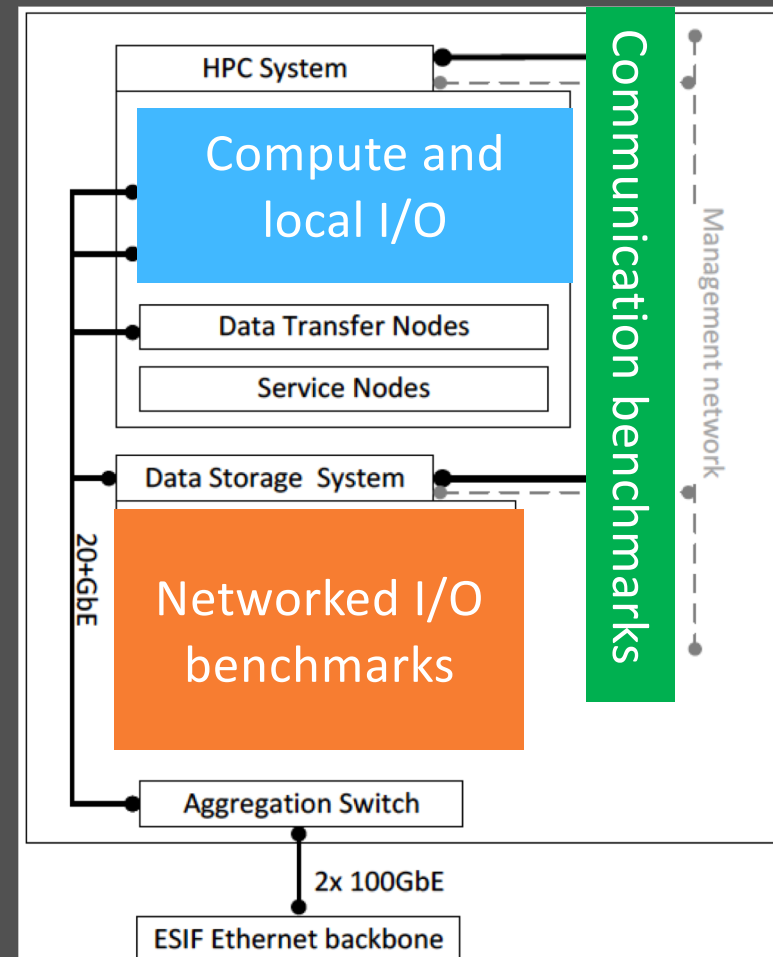
# Hints to Architect



- **Things we noticed then**
  - Skewed toward throughput
  - Certain workloads memory-intensive (256GB nodes)
  - Sometimes local scratch disk handy
- **Trends we saw coming**
  - Accelerators
  - Machine Learning

# Rough Motivating Architecture

- **Biased toward x86\_64**
  - standard nodes (1.5 GB DRAM/core), ~200 GB local persistent storage
  - Large memory compute
  - GPU + large memory compute
- **Shared parallel filesystem**
- **Shared utility filesystem**
- **High performance network with utility GbE connections**





# ESIF-HPC-2 Benchmark Suite

---

Why Assemble a Benchmark Suite?

# Why Benchmarks for Us?

- Quantitative performance **discriminator** across potential systems
- Enabling **responsive design**
- **Validating** delivered system
- Quantifying burst **reliability at speed**
- **Continuous verification** in production
- Detailed understanding of requirements to achieve performance
- **Setting expectations** for future system

# Why Benchmarks for Others?

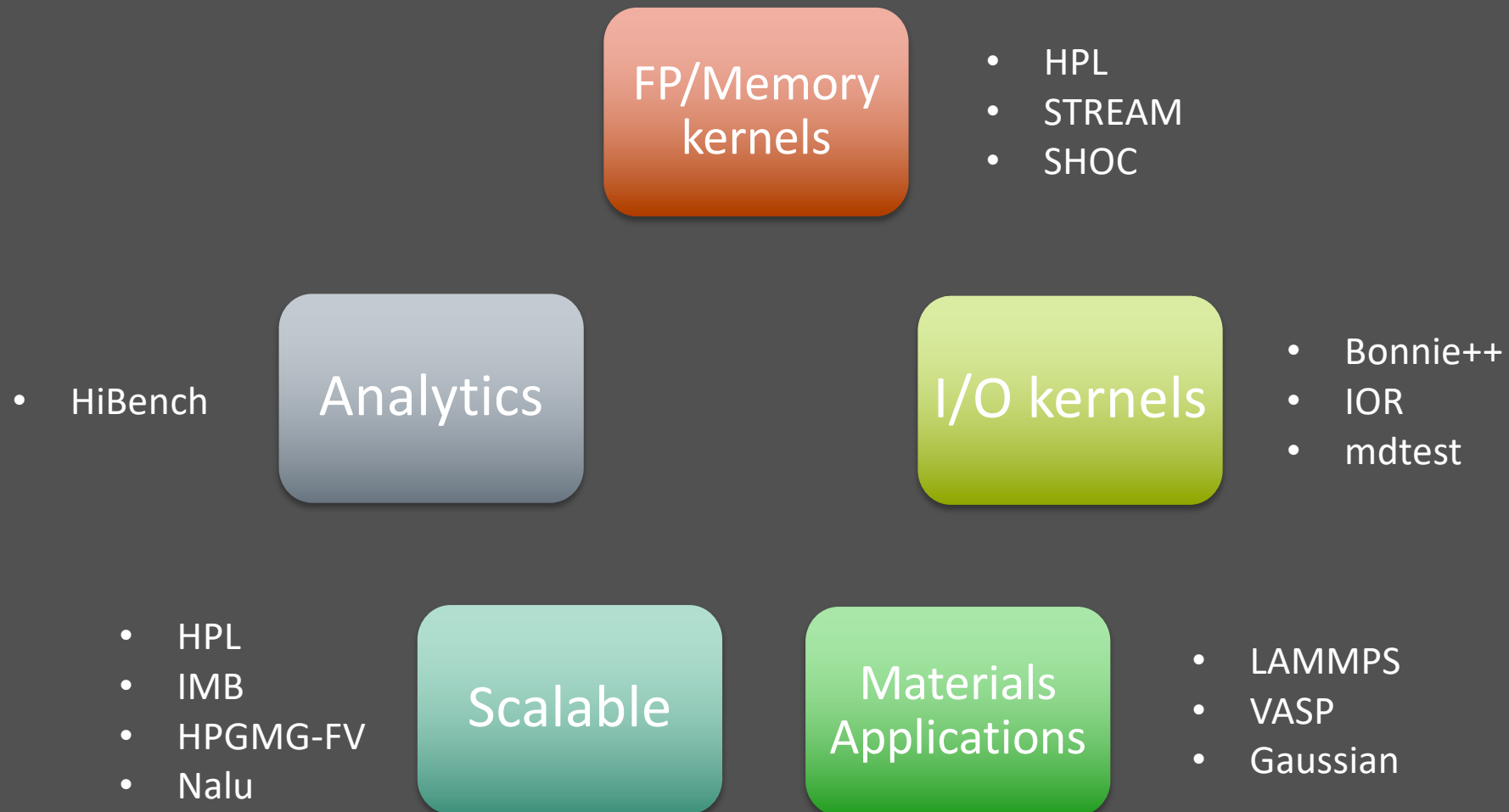
- Procurements range from vanilla to Devil's Breath Carolina Reaper Pepper <https://www.mentalfloss.com/article/51703/12-strange-real-ice-cream-flavors>
  - If your architectural constraints are similar, why reinvent?
- Standardization—what are commonalities?
- A starting point for newbs
- A historical record (if abuse GitHub)

# ESIF-HPC-2 Benchmark Suite


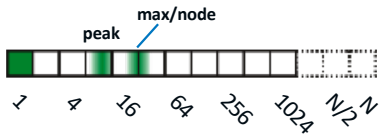
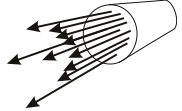

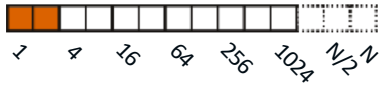
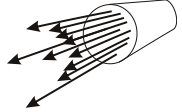

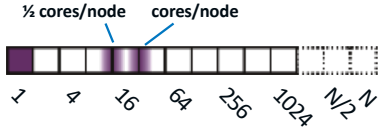
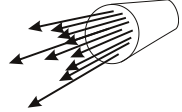

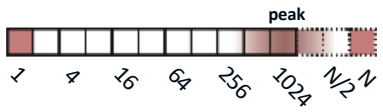
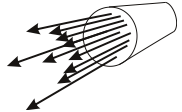

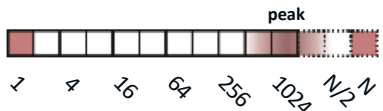

---

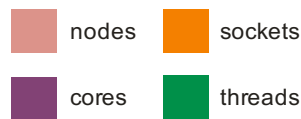
What's in the Suite, How it was built,  
and How it was run

# High-Level Grouping

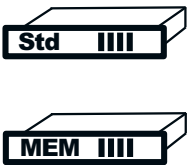
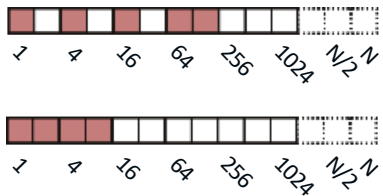


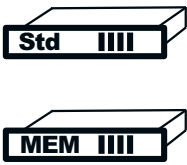



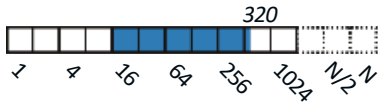



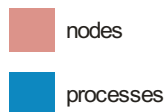
# Kernels

STREAM			<ul style="list-style-type: none"> <li>Triad</li> <li>Default &amp; 60% DRAM</li> </ul>	
SHOC			<ul style="list-style-type: none"> <li>BusSpeed tests (Level 0)</li> <li>Triad (Level 1)</li> </ul>	
Bonnie++	 +login +service		<ul style="list-style-type: none"> <li>Default transfer settings</li> <li>Local, HFS &amp; PFS</li> <li>SR, SRW, SW</li> </ul>	
IOR			<ul style="list-style-type: none"> <li><math>\geq 1.5 \times</math> mem/node, 80% full</li> <li>PFS + HFS; POSIX and MPI I/O</li> <li>file/process and shared file</li> </ul>	
mdtest			<ul style="list-style-type: none"> <li>1 or 1048576 files, single/multiple directories</li> <li>Offeror reports best # ranks</li> <li>Create/stat/remove</li> </ul>	 rate ( $s^{-1}$ )


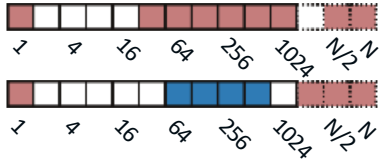



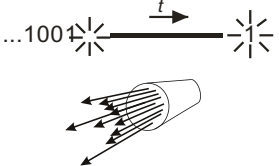




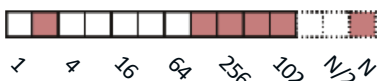



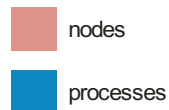
# Materials Applications

LAMMPS			<p>35% LiCl solution</p> <ul style="list-style-type: none"> <li>3 sizes: <math>7 \times 10^5</math>, <math>6 \times 10^6</math>, <math>4.8 \times 10^7</math> atoms</li> </ul>	 #steps  loop timesteps/s
Gaussian			<p><math>\omega</math>B97X SP Mn-aquo complex</p> <ul style="list-style-type: none"> <li>175 <math>e^-</math></li> <li>520 BF</li> </ul>	 wallclock
VASP			<p>Two components</p> <ul style="list-style-type: none"> <li>Semiconductor <math>\text{Cu}_4\text{In}_4\text{Se}_8</math> GW(10-10-5)</li> <li>Catalysis <math>\text{Ag}_{504}\text{C}_4\text{H}_{10}\text{S}</math> GGA(<math>\Gamma</math>)</li> </ul>	 wallclock



# Scalable

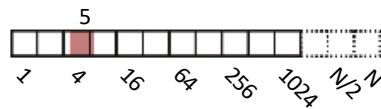
HPL			<p>Offeror tunes ranks/node, thread/node, N, NB, P, and Q for optimal performance</p>	
IMB			<ul style="list-style-type: none"> <li>0, 64 kB, 0.5 MB, 4MB</li> <li>9 tests, incl. PingPong, OB Barrier, Uni/Bi band, and Alltoall</li> </ul>	
HPGMG-FV			<p><math>2^7</math>-unit box, 8 boxes/rank</p>	
Nalu			<ul style="list-style-type: none"> <li>256 mesh</li> <li>scaling + throughput tests</li> </ul>	





# Analytics

HiBench



nodes

- Hadoop & Spark
- Wordcount, Sort, Bayes, K-means, DFS I/O Enhanced
- “gigantic” ( $10^{10}$ - $10^{11}$  B)

 B/s  wallclock

# Responses

- Three classes
  - Spreadsheet response, where the numbers go;
  - Text response, where the words go; and,
  - File response, where the results and inputs go
- Not integral to the benchmarks, but may be useful to structure runs and records

# Process

- One person per benchmark
- One GitHub repo per benchmark
  - Internal GitHub allows freedom to experiment
  - Can pull independently
  - Change requests, etc. built in
- Third-party testing is a simple branch
  - Branch, change README.md, add permissions to tester

# ESIF-HPC-2 Benchmark Suite

---

Benchmark Coverage

# Benchmarks in Space

- Think of benchmarks as occupying points in a space
- Considerations are subspaces, with multiple dimensions each
- Allows us to start formalizing what aspects we're testing

Subspace	Type	Dimensions
Hardware subsystem	Categorical	processor, memory, storage, network
Parallel scope	Ordinal	serial, MT/MP <sub>1</sub> single node, multi-node, many-node/scalable
Software scope	Ordinal	kernel, mini-application, full application, workflow
Degree of task coupling	Ordinal	loose, medium, tight
Data transfer	Categorical	Cache-core, memory-core, LFS-memory, NFS-memory, PFS-memory, external-memory, memory-memory <sub>2</sub>
Performance type	Categorical	Maximum application, sustained
Algorithms	Categorical	See [7], [8]
Interactive productivity	Categorical	Job scheduler delay, GUI latency, framework support

# Benchmark Vectors

Benchmark	Hardware subsystem				Parallel scope			Software scope			Task coupling			Data transfer					Performance		Algorithms																	
	Processor	memory	storage	network	serial	MT/Mp single node	multi-node scalable	kernel	mini-application	full application workflow	loose	medium	tight	cache-core	memory-core	LFS-memory	NFS-memory	PFS-memory	external-memory	memory-memory	maximum	sustained	SG	UG	Spectral	DLA	SLA	N-body	MC	CL	GT	GM	FSM	Dp	BnB	Interactive productivity		
STREAM Triad	0		0	0	0		0	0		0	0	0		0		0	0	0	0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	
HPL		0	0	0	0	0			0	0		0	0		0	0	0	0	0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	
SHOC Triad	0			0	0		0	0				0	0		0	0	0	0	0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	
Bonnie++	0				0		0	0				0	0		0							0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	
IOR	0	0		0	0							0	0		0		0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
mdtest	0	0		0	0							0	0		0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
IMB	0	0	0		0			0	0	0				0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
HPGMG-FV	0		0		0	0	0		0		0		0			0	0	0	0		0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	
Nalu					0	0	0		0				0		0	0		0	0		0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	
VASP			0		0		0		0	0		0			0	0	0		0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	
LAMMPS					0			0	0		0		0		0	0		0	0		0	0	0	0	0		0		0	0	0	0	0	0	0	0	0	
Gaussian			0	0	0		0	0		0		0			0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	
HiBench	0			0	0	0	0		0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Totals	5	8	6	5	0	5	7	7	8	1	4	0	7	4	3	1	7	1	1	3	0	6	8	5	1	1	2	5	2	1	0	0	0	0	0	0	0	0

# Conclusions

- ESIF-HPC has an eclectic and evolving mix of applications, job sizes, and mission requirements to design against
- The ESIF-HPC-2 benchmark suite – grab'n'go set of tests
- Mix of kernel, application, data-centric, and scalable
- Suite used standard development tools to standardize workflows
- Idea of benchmarks as a space allows one to assess coverage
- <https://github.com/NREL/ESIFHPC2>