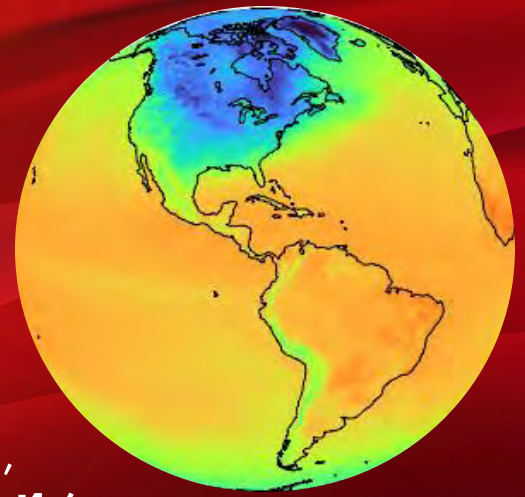
BOOSTING EARTH SYSTEM MODEL OUTPUTS AND SAVING PETABYTES IN THEIR STORAGE USING EXASCALE CLIMATE EMULATORS



Sameh Abdulah[1], Allison H. Baker[2], George Bosilca[3], **Qinglei Cao**[4], Stefano Castruccio[5], Marc G. Genton[1], David E. Keyes[1], **Zubair Khalid**[1,6], Hatem Ltaief[1], Yan Song[1], Georgiy L. Stenchikov[1], and Ying Sun[1]

[1] Extreme Computing & Statistics & Earth Science, King Abdullah University of Science and Technology, KSA

[2] Computational and Information Sciences Lab, NSF National Center for Atmospheric Research, USA

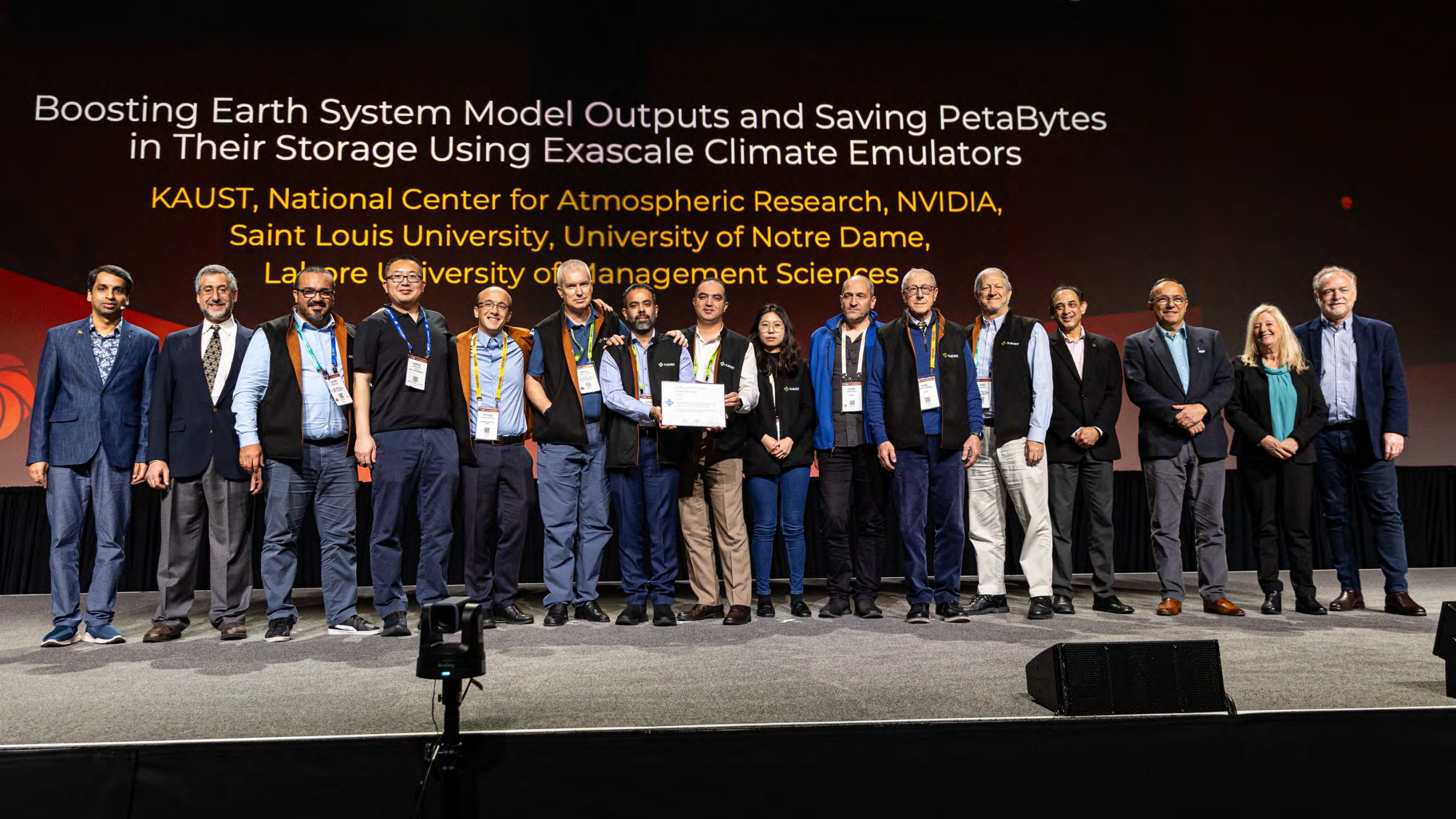
[3] NVIDIA, USA

[4] Department of Computer Science, Saint Louis University, USA

[5] Department of Applied and Computational Mathematics and Statistics, University of Notre Dame, USA

[6] Department of Electrical Engineering, Lahore University of Management Sciences, Pakistan

SC25 St. Louis, MO | hpc ignites.


Boosting Earth System Model Outputs and Saving PetaBytes
in Their Storage Using Exascale Climate Emulators
KAUST, National Center for Atmospheric Research, NVIDIA,
Saint Louis University, University of Notre Dame,
Lahore University of Management Sciences

OUTLINE

- PART 1:

  Earth System Models: Computation and Storage Challenges and Motivation

- PART 2:

  Exascale Climate Emulators – Design Overview

- PART 3:

  HPC Innovations, Solutions, and Performance

PART 1

Earth System Models: Computation and Storage Challenges and Motivation

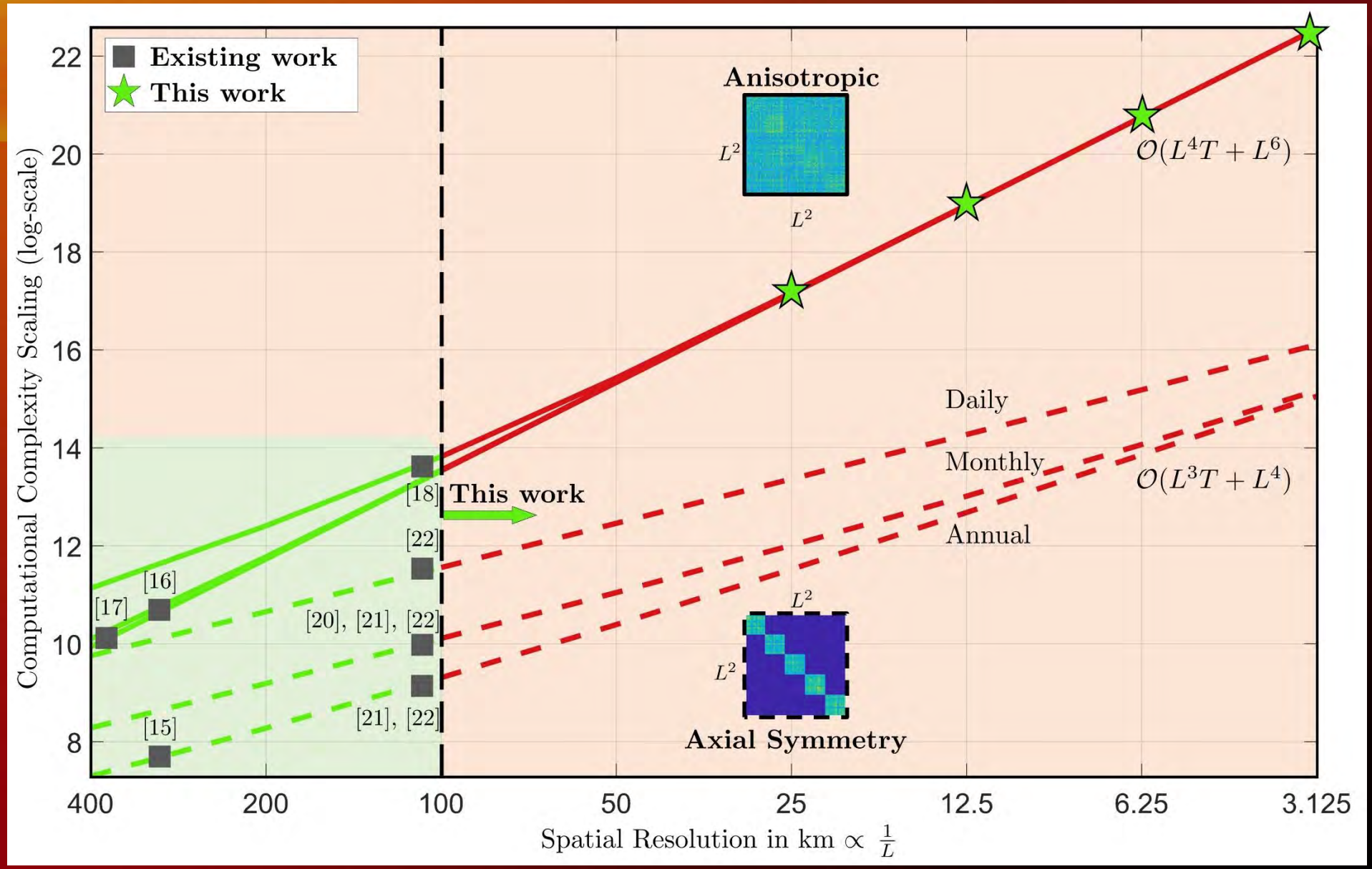
MOTIVATIONS & CHALLENGES

- Our understanding of the climate system relies on **state-of-the-art Earth System Models (ESMs)** based on PDEs and run on supercomputers (very few runs can be afforded!)
- ESMs play a fundamental role in the **Intergovernmental Panel on Climate Change (IPCC) sixth assessment report** (AR6) to forecast warming across various emission scenarios
- The latest **Coupled Model Intercomparison Project** (CMIP6) supports detailed comparisons of ESMs (generated ~28 Petabytes data from 45 modeling institutes)
- **Computational demands and petabyte-scale storage requirements/costs** for ESMs continue to escalate as the climate community progresses toward **ultra-high-resolution** simulations
- **Simulations at "global storm-resolving" scales** are needed to understand better how weather and extremes will be affected by climate change

CLIMATE EMULATORS

- Climate emulators are **stochastic models designed to mimic ESM behavior** using simulated data from a few runs of the ESM
- Climate emulators can **quickly generate multiple emulations** of the output of an ESM, which is **crucial for quantifying the uncertainties** in future climate projections
- Climate emulators are designed to **complement and boost the usefulness of ESMs**
- Current global climate emulators **have not yet attained a spatial resolution finer than 100 km**
- Current global climate emulators **have not yet attained a temporal resolution finer than daily (and only then with an assumption of axial symmetry)**
- Current global climate emulators **without assumption of axial symmetry have not yet attained a temporal resolution finer than annually**

CLIMATE EMULATORS: EXISTING WORK & CHALLENGES



SUMMARY OF CONTRIBUTIONS (1/2)

- Developed and validated **our own climate emulator** (Song, Khalid, Genton, 2024, JASA)
- Our **Exascale Climate Emulator surpasses** existing climate emulators **by a factor of 245,280X** (28X in space and 8,760X in time)
- Addresses limitations of existing emulators – **spherical harmonic transform (SHT) to model anisotropic interactions**
- Our Exascale Climate Emulator can emulate up to **54.5 million spatial locations** across the globe with an **ultra-high spatial resolution of up to 0.034° (3.5 km) at an hourly resolution.** This equates to **477 billion data points** for a single year of emulation
- Virtually **saving infinite number of Petabytes of storage and cost** via on-demand climate data emulations
- **Breaking the ultra-high-resolution barrier** of climate emulators for **advancing climate research and policy making**

KAUST Shaheen III

SUMMARY OF CONTRIBUTIONS (2/2)

- **Leading the way to sustainable climate modeling** on supercomputers via **PaRSEC runtime system** orchestrating **mixed-precision computational tasks**
- Large-scale execution and performance demonstration on systems equipped with **four different GPU accelerators**:

0.375 EFlop/s on 3,072 nodes (18,432 NVIDIA V100 GPUs) of Summit

0.243 EFlop/s on 1,024 nodes (4,096 NVIDIA A100 GPUs) of Leonardo

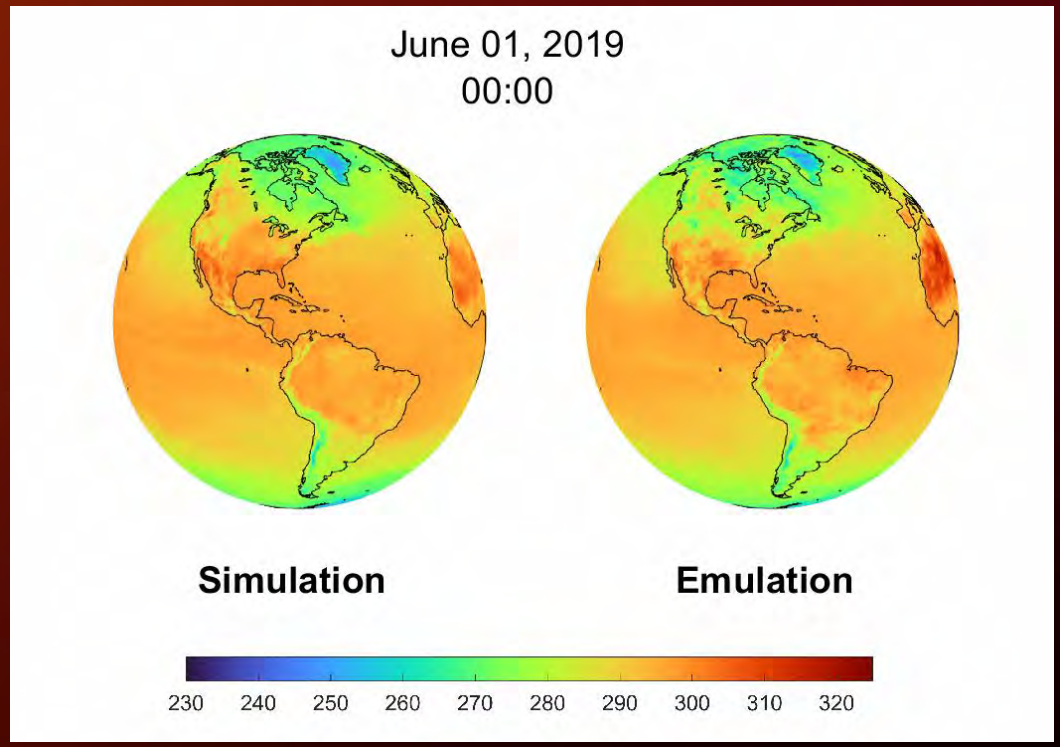
0.739 EFlop/s on 1,936 nodes (7,744 NVIDIA GH200 GPU superchips) of Alps

0.976 EFlop/s on 9,025 nodes (36,100 AMD MI250X GPUs) of Frontier

- Excellent weak scaling efficiency, up to **72% strong scaling efficiency** with up to 12,288 V100 GPUs on Summit

ESM DATA

- **ERA5 surface temperature dataset** at 25 km spatial resolution (1,038,240 locations)
- **318 billion hourly data** spanning 35 years (1988-2022)
- **31 billion daily data** spanning 83 years (1940-2022)



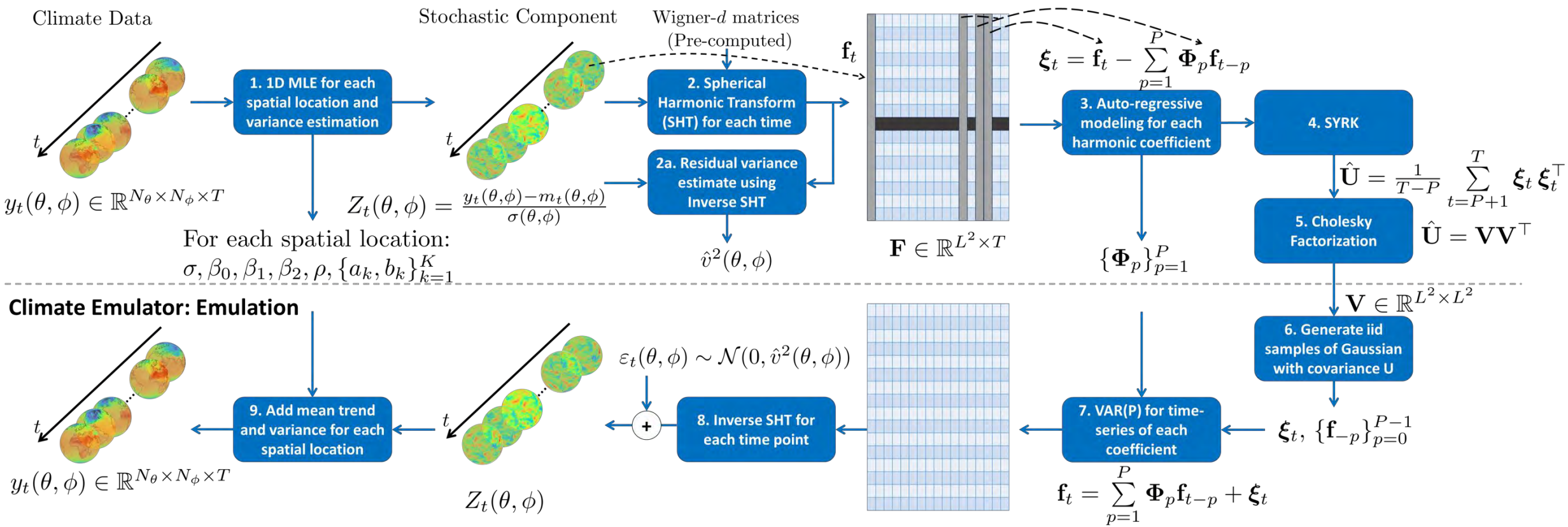
- Upscaling to **ultra-high spatial resolution** of 3.5 km (54,486,360 locations)
- **477 billion hourly data per year**

PART 2

Exascale Climate Emulators – Design Overview

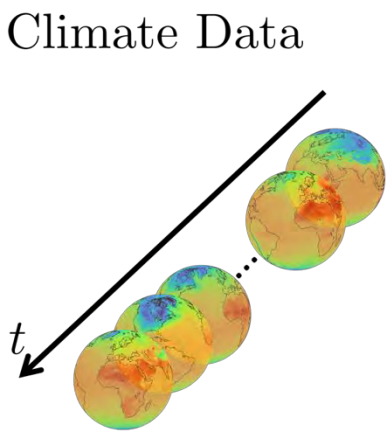
CLIMATE EMULATOR DESIGN

Big Picture:



CLIMATE EMULATOR DESIGN

Stage 1: Mean Trend Removal and Variance Normalization:



$$y_t(\theta, \phi) \in \mathbb{R}^{N_\theta \times N_\phi \times T}$$

$$y_t^{(r)}(\theta_i, \phi_j) = m_t(\theta_i, \phi_j) + \sigma(\theta_i, \phi_j) Z_t^{(r)}(\theta_i, \phi_j)$$

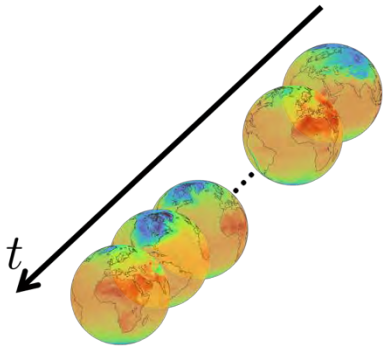
$$m_t = \beta_0 + \beta_1 x_{\lceil t/\tau \rceil} + \beta_2 (1-\rho) \sum_{s=1}^{\infty} \rho^{s-1} x_{\lceil t/\tau \rceil - s} + \sum_{k=1}^{K} \left\{ a_k \cos\left(\frac{2\pi t k}{\tau}\right) + b_k \sin\left(\frac{2\pi t k}{\tau}\right) \right\}$$

Mean trend

CLIMATE EMULATOR DESIGN

Stage 1: Mean Trend Removal and Variance Normalization:

Climate Data



$y_t(\theta, \phi) \in \mathbb{R}^{N_\theta \times N_\phi \times T}$

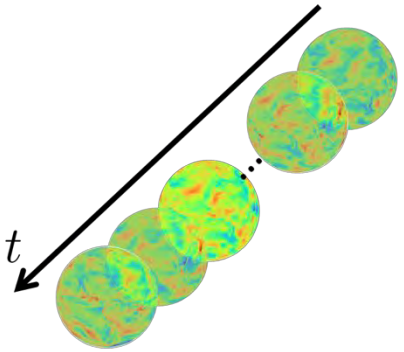
1. 1D MLE for each spatial location and variance estimation

For each spatial location:

$\sigma, \beta_0, \beta_1, \beta_2, \rho, \{a_k, b_k\}_{k=1}^{K}$

(Estimated Parameters)

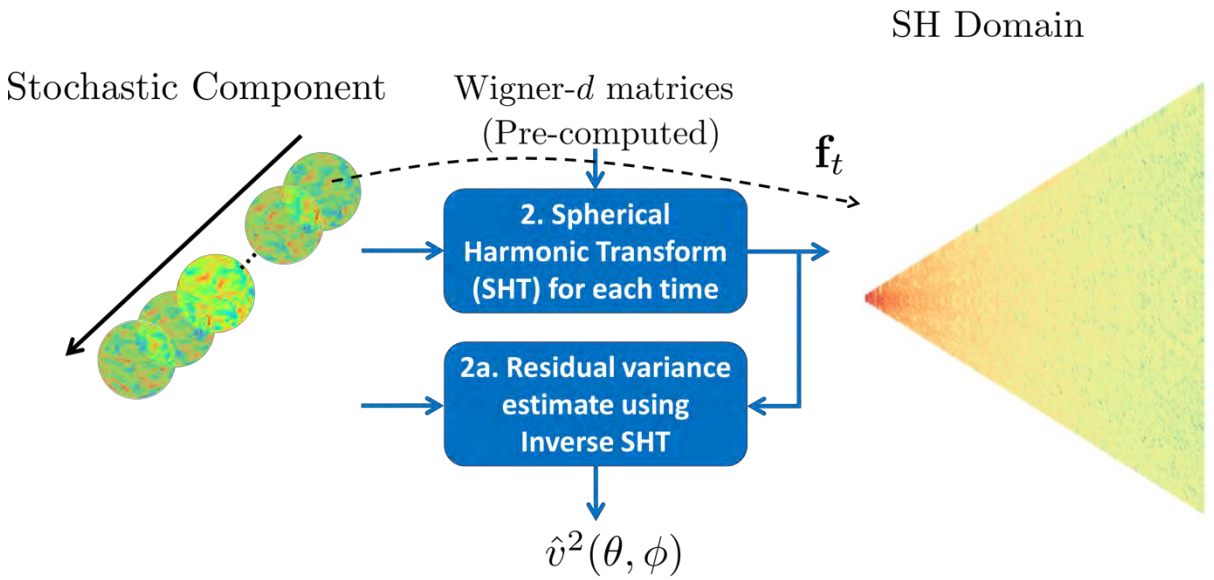
Stochastic Component



$$Z_t(\theta, \phi) = \frac{y_t(\theta,\phi) - m_t(\theta,\phi)}{\sigma(\theta,\phi)}$$

CLIMATE EMULATOR DESIGN

Stage 2 to 4: Modeling in Spherical Harmonic Domain – Anisotropic Covariance Matrix



(Error in SH representation)

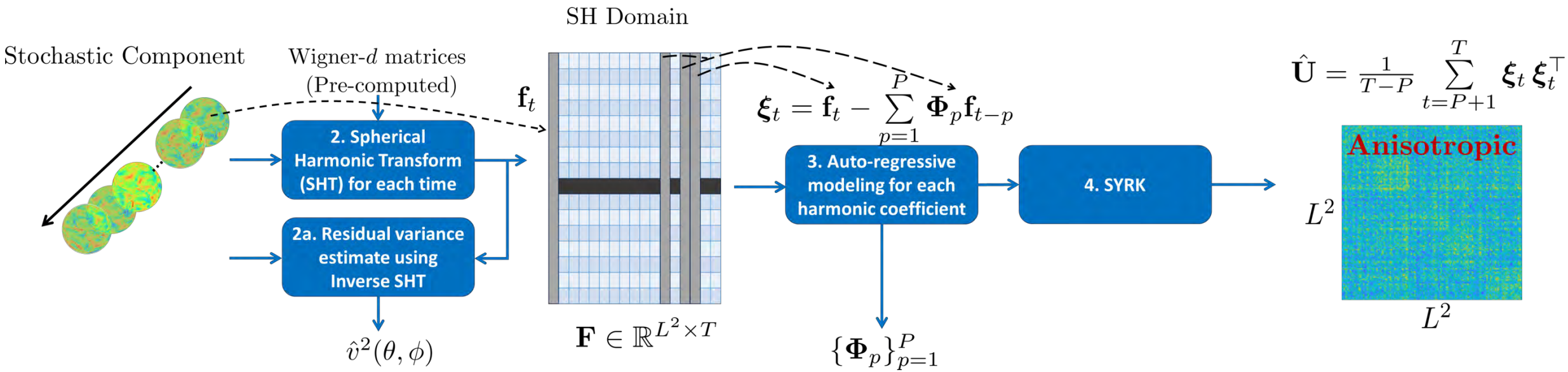
$\mathbf{f}_t \in \mathbb{R}^{L^2}$

– L is the spherical harmonic band-limit

– P is the order of AR model

CLIMATE EMULATOR DESIGN

Stage 2 to 4: Modeling in Spherical Harmonic Domain – Anisotropic Covariance Matrix



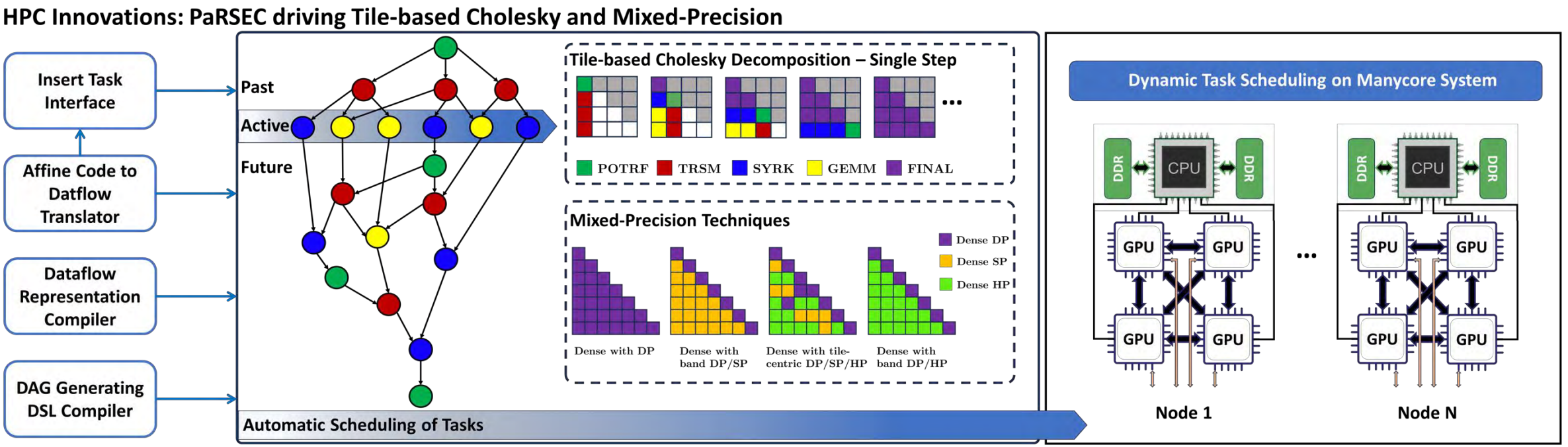
$\mathbf{f}_t \in \mathbb{R}^{L^2}$

– L is the spherical harmonic band-limit

– P is the order of AR model

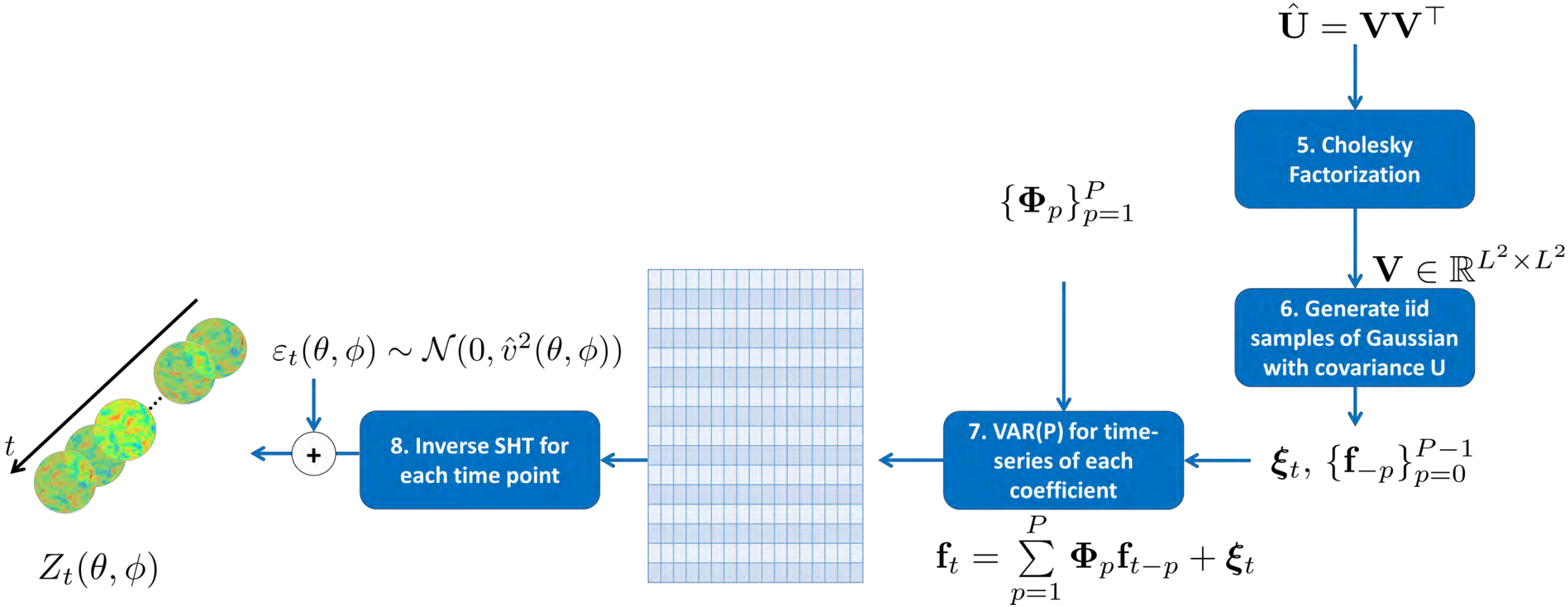
CLIMATE EMULATOR DESIGN

Stage 5: Cholesky Factorization:



CLIMATE EMULATOR DESIGN

Stage 5 to 8: Generating Statistically Consistent Stochastic Component:



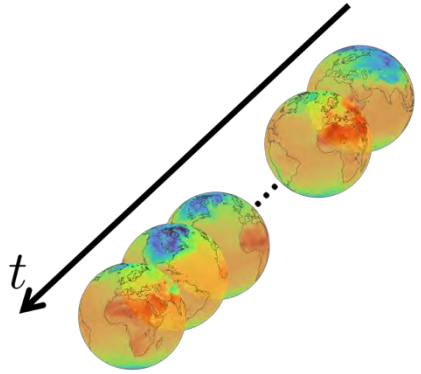
CLIMATE EMULATOR DESIGN

Stage 9: Incorporating Mean Trend and Spatial Variance:

For each spatial location:

$$\sigma, \beta_0, \beta_1, \beta_2, \rho, \{a_k, b_k\}_{k=1}^{K}$$

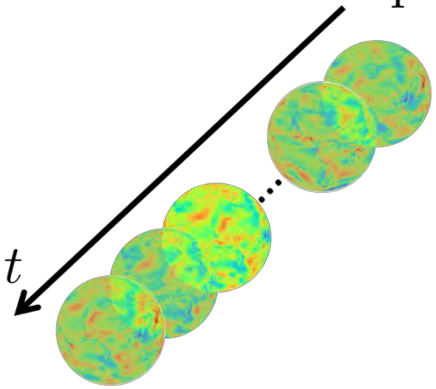
Climate Emulation



$$y_t(\theta, \phi) \in \mathbb{R}^{N_\theta \times N_\phi \times T}$$

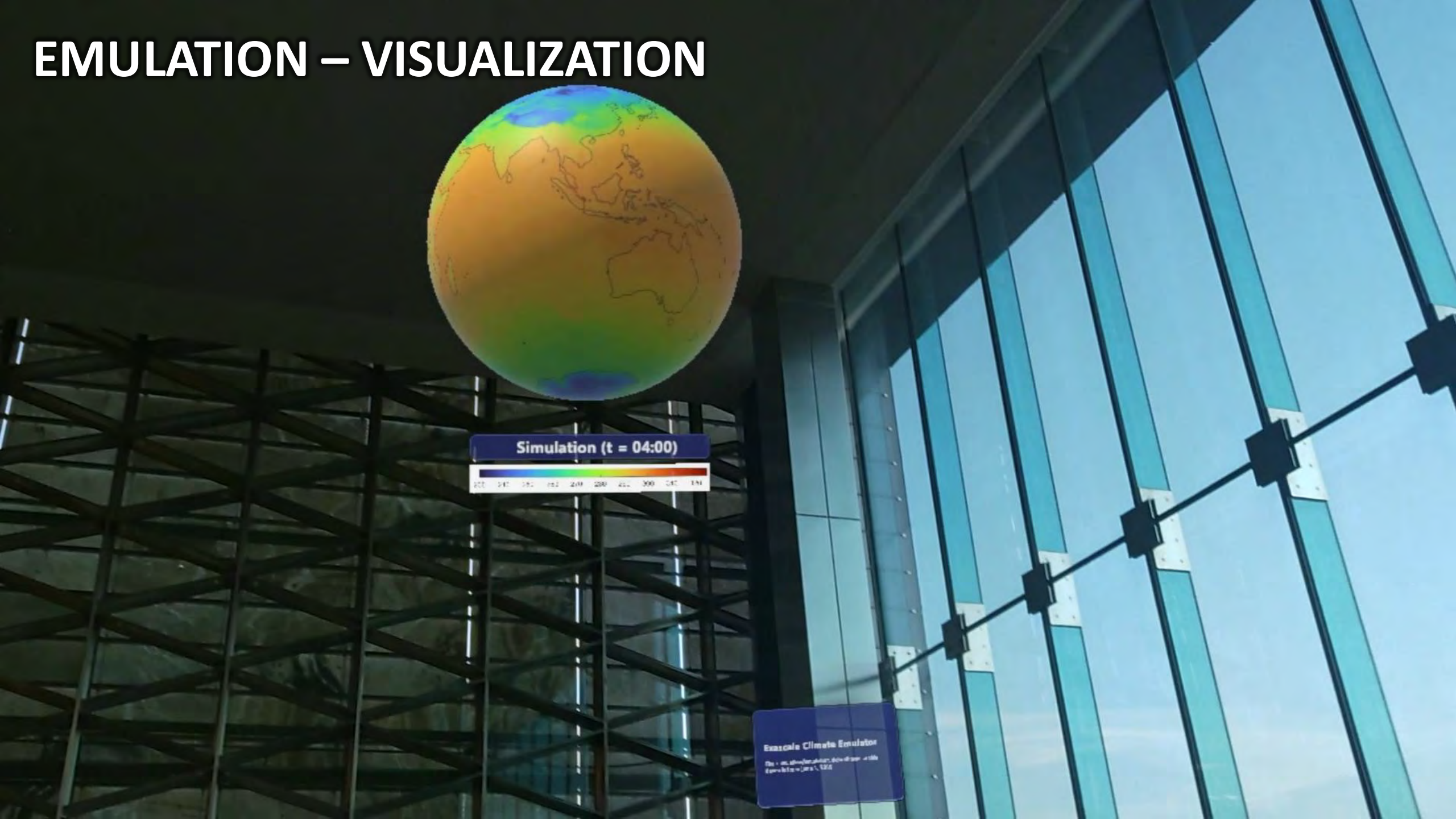
9. Add mean trend and variance for each spatial location

Stochastic Component



$$Z_t(\theta, \phi)$$

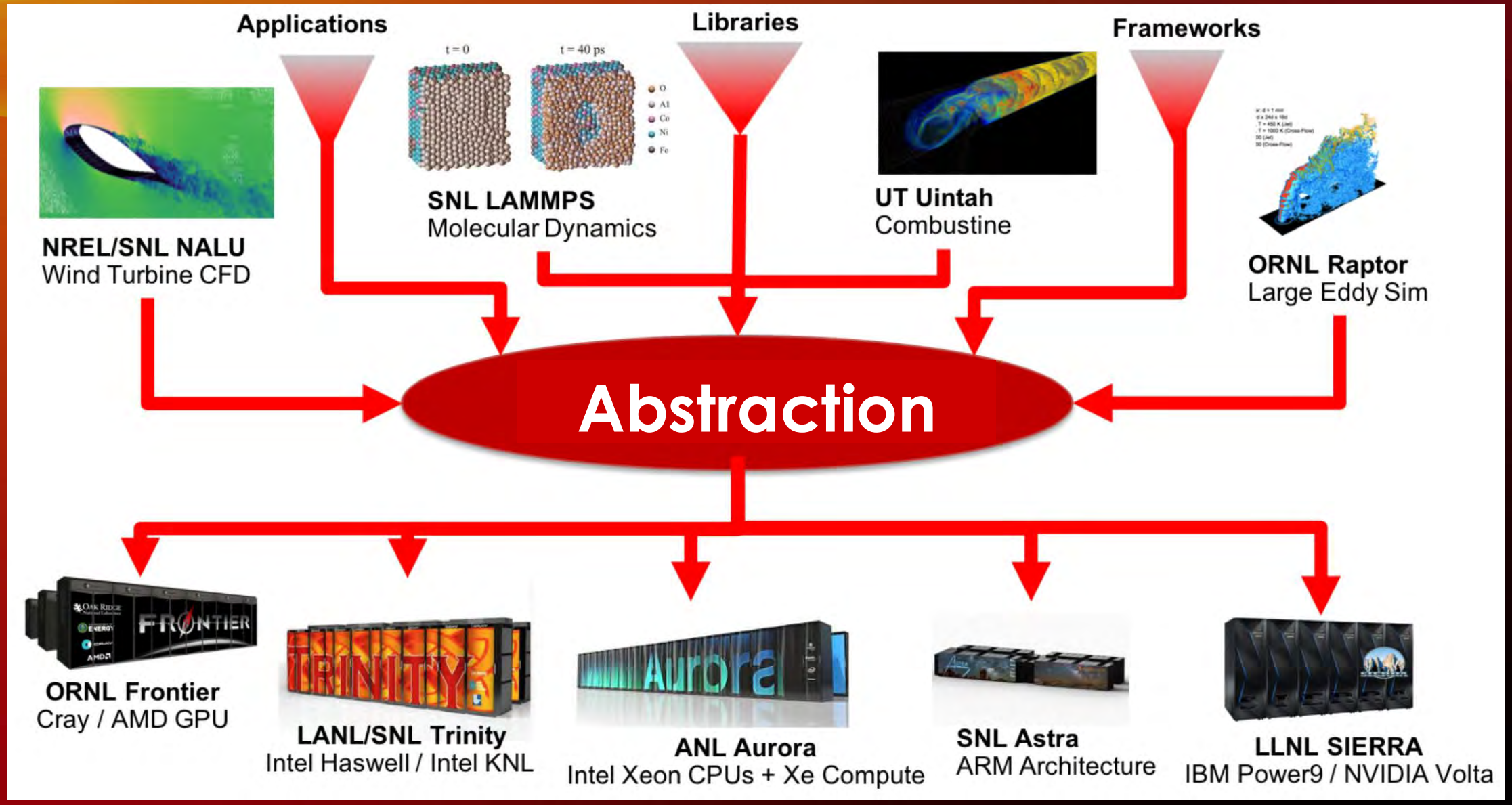
EMULATION – VISUALIZATION



PART 3

HPC Innovations, Solutions, and Performance

COMPLEXITY IN HARDWARE AND SOFTWARE

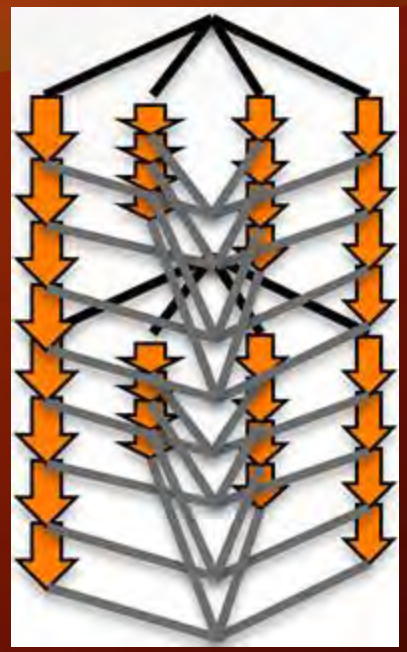


PROGRAMMING PARADIGMS

BSP and early message passing

MPI + X

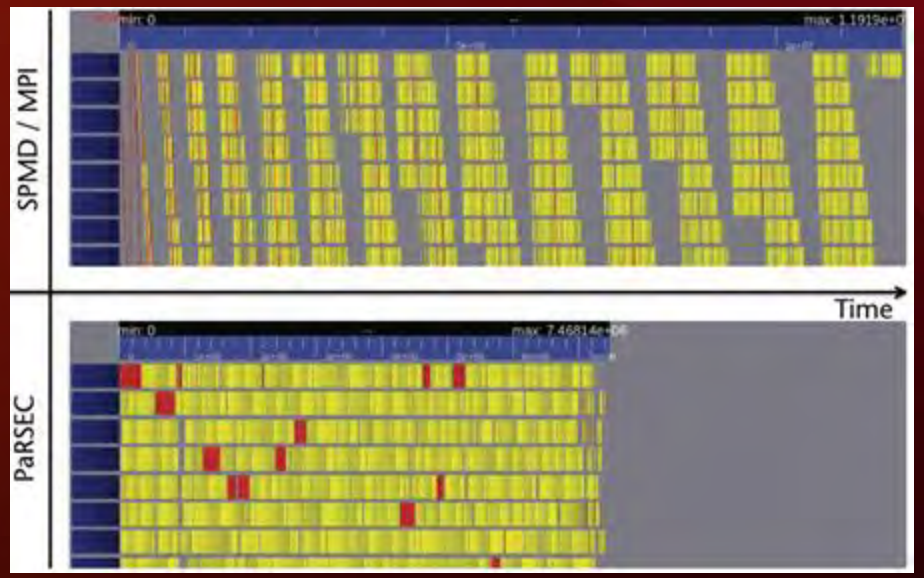
MPI + X + Y



Task-base runtime

- Harder than sequential programming
- Users need to express parallelism with minimum synchronization points
- Managing shared memory, distributed memory and heterogeneous architectures

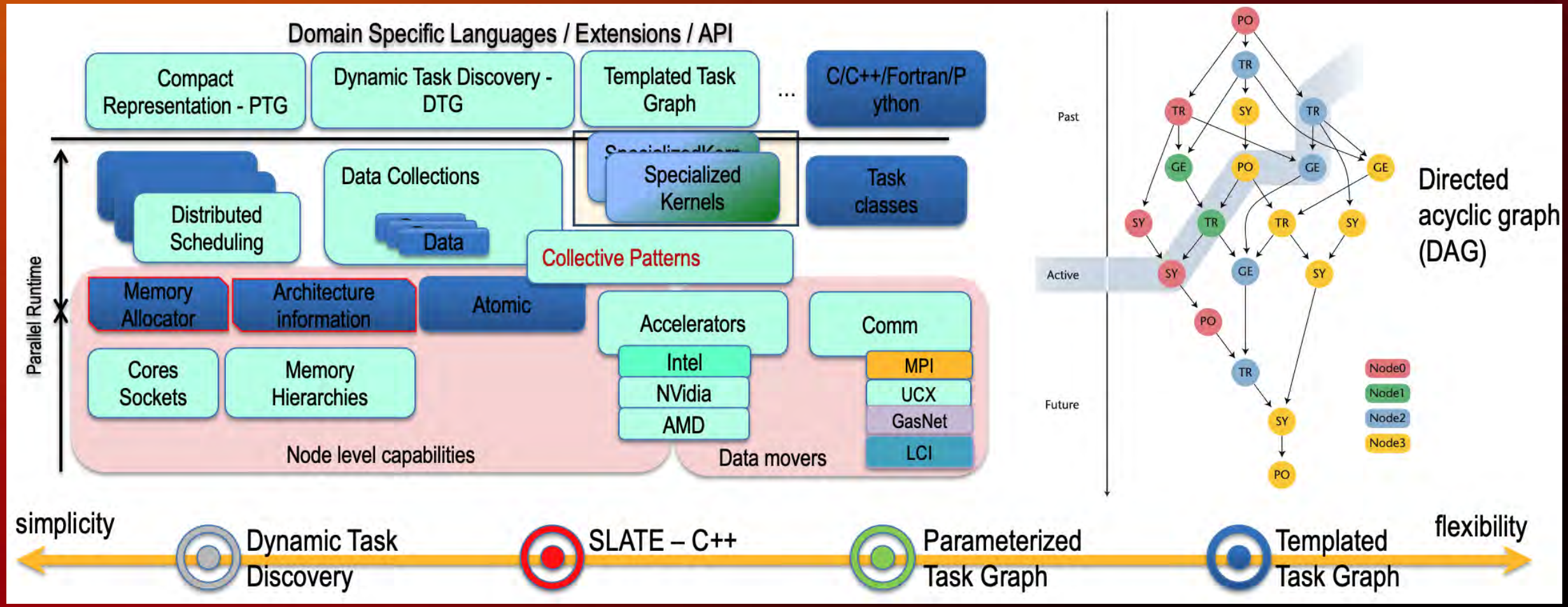
Many task-based runtime: OmpSs, OpenMP, StarPU, Legion, **PaRSEC** ...



Comparison of execution traces for the same algorithm using the single-program, multiple data message passing interface (SPMD/ MPI) programming model and the dataflow model

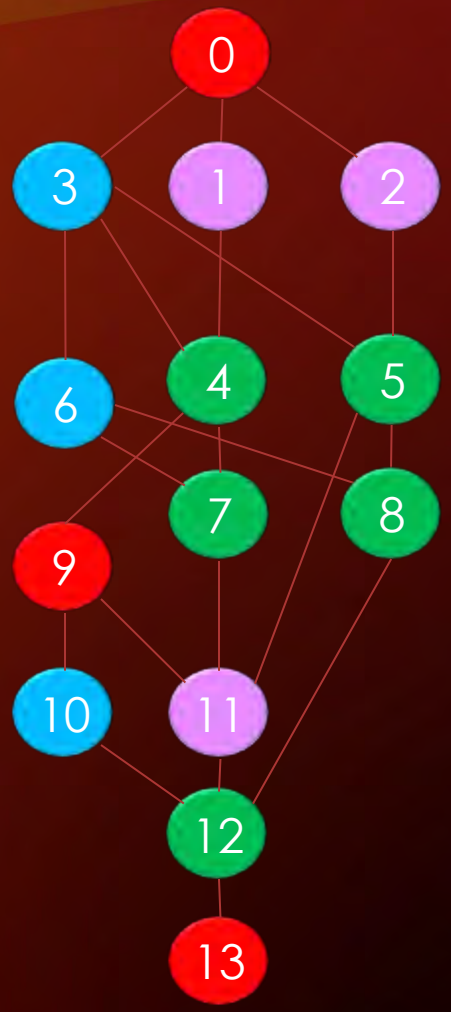
PARSEC (HTTPS://GITHUB.COM/ICLDISCO/PARSEC)

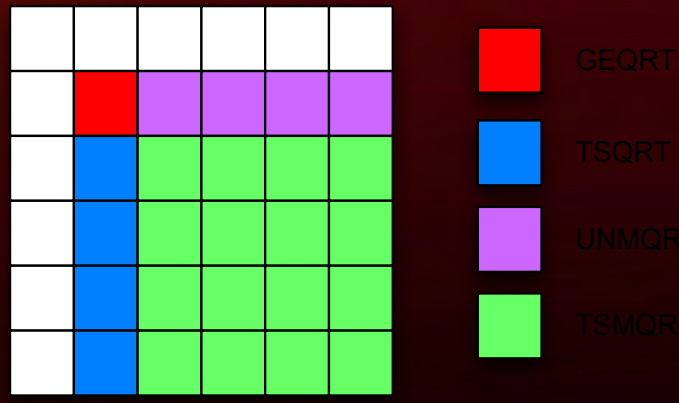
A generic runtime system for asynchronous, architecture aware scheduling of fine-grained tasks on distributed many-core heterogeneous architectures.



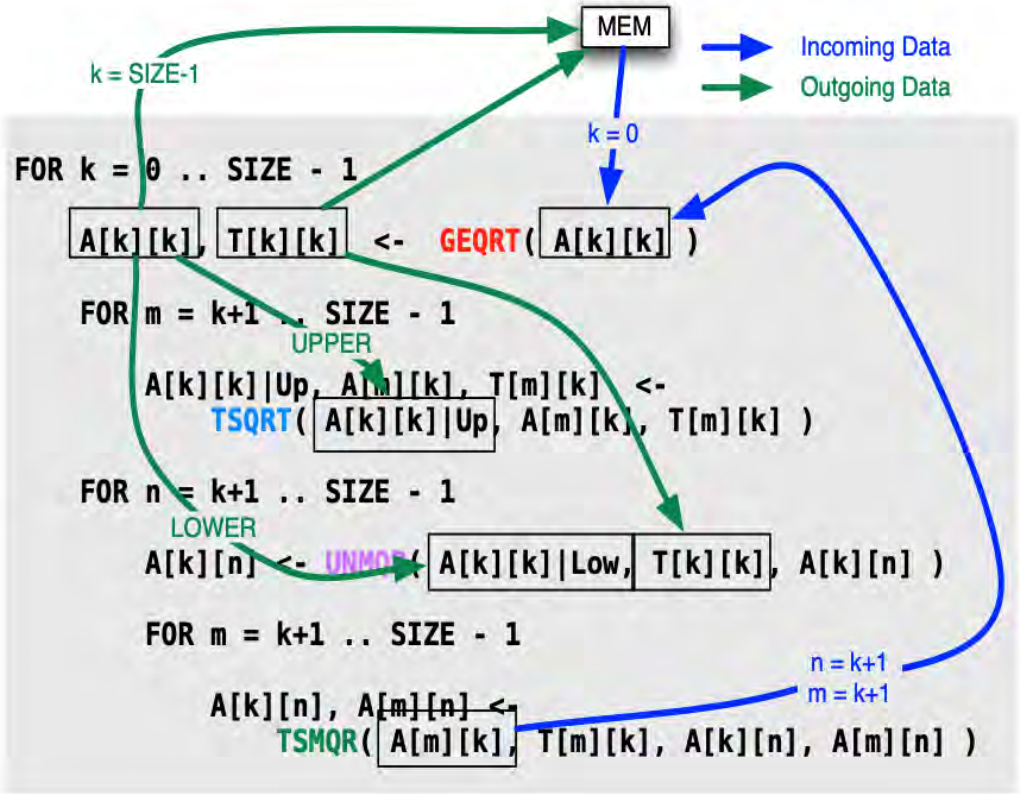
DTD: insert_task

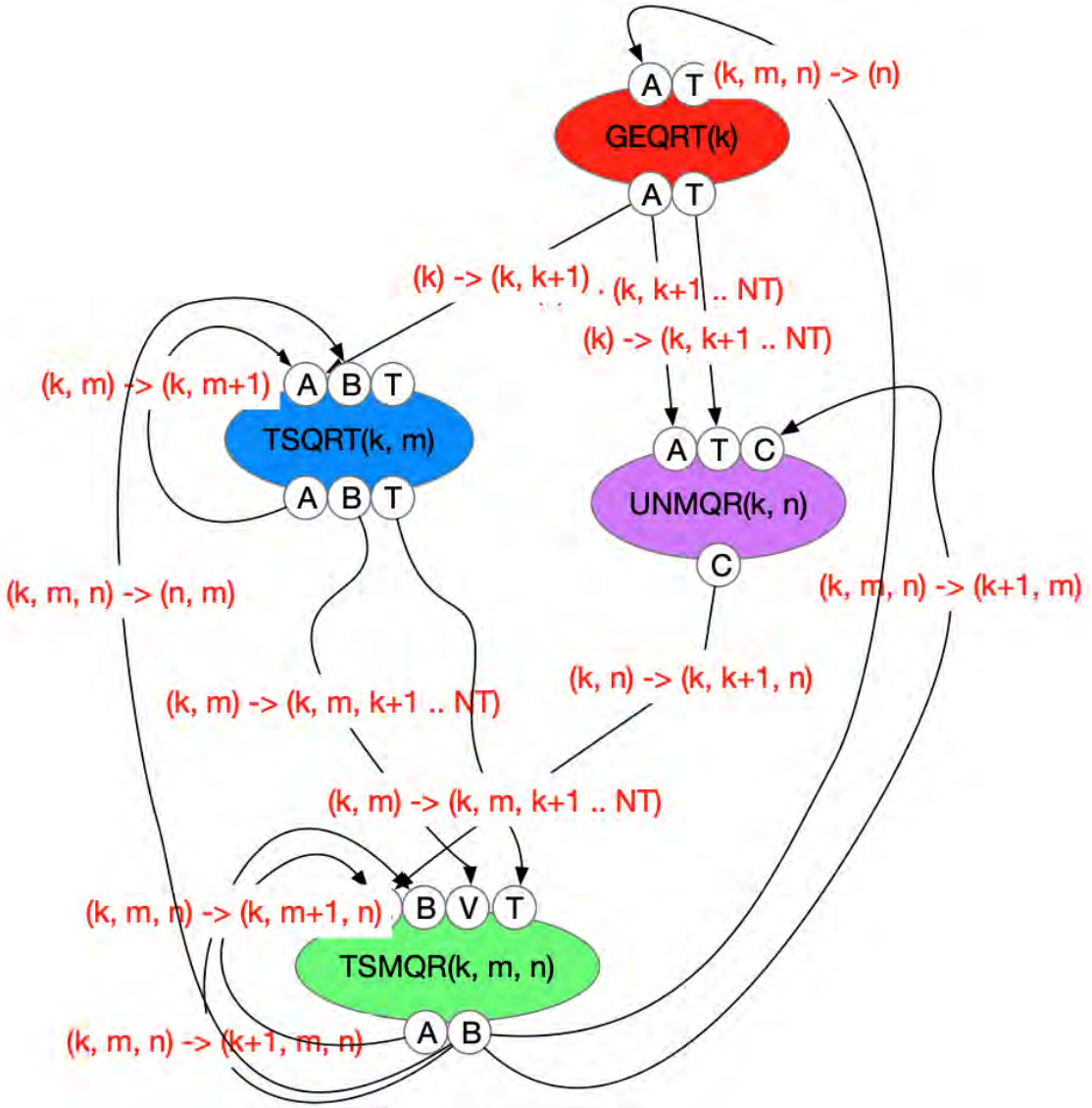
```
for( k = 0; k < SIZE; k++ ) {
    parsec_insert_task( "GEQRT",
                DATA_OF(A, k, k), INOUT|AFFINITY,
                DATA_OF(T, k, k), OUTPUT|TILE_RECT)

    for( n = k+1; n < SIZE; n++ )
        parsec_insert_task( "UNMQR",
                DATA_OF(A, k, k), INPUT|TILE_L,
                DATA_OF(T, k, k), INPUT|TILE_RECT,
                DATA_OF(A, k, n), INOUT|AFFINITY)

    for( m = k+1; m < SIZE; m++ ) {
        parsec_insert_task( "TSQRT",
                DATA_OF(A, k, k), INOUT|TILE_U,
                DATA_OF(A, m, k), INOUT|AFFINITY,
                DATA_OF(T, m, k), OUTPUT|TILE_RECT)

        for( n = k+1; n < SIZE; n++ ) {
            parsec_insert_task( "TSMQR",
                    DATA_OF(A, k, n), INOUT,
                    DATA_OF(A, m, n), INOUT|AFFINITY,
                    DATA_OF(A, m, k), INPUT,
                    DATA_OF(T, m, k), INPUT|TILE_RECT)
        }
    }
}
```





DSL: THE PARAMETERIZED TASK GRAPH (.JDF)





Used in this project

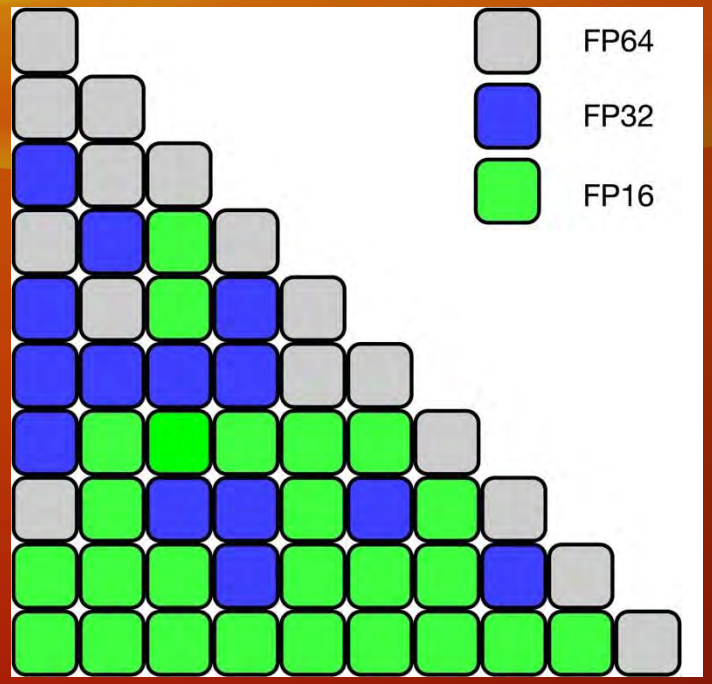
MOTIVATIONS FOR MIXED PRECISIONS

- Many matrices arising in applications have blocks of relatively small norm and can be replaced with reduced precision.
- Computational: faster time to solution
  - ✓ lower energy consumption and higher performance, especially by exploiting heterogeneity

Peak Performance in TF/s	V100 NVLink	A100 NVLink	H100 SXM
FP64	7.8	9.7	34
FP32	15.7	19.5	67
FP64 Tensor Core	16x	19.5	67
TF32 Tensor Core		156 16x	494.7 15x
FP16 Tensor Core	125	312	989.4

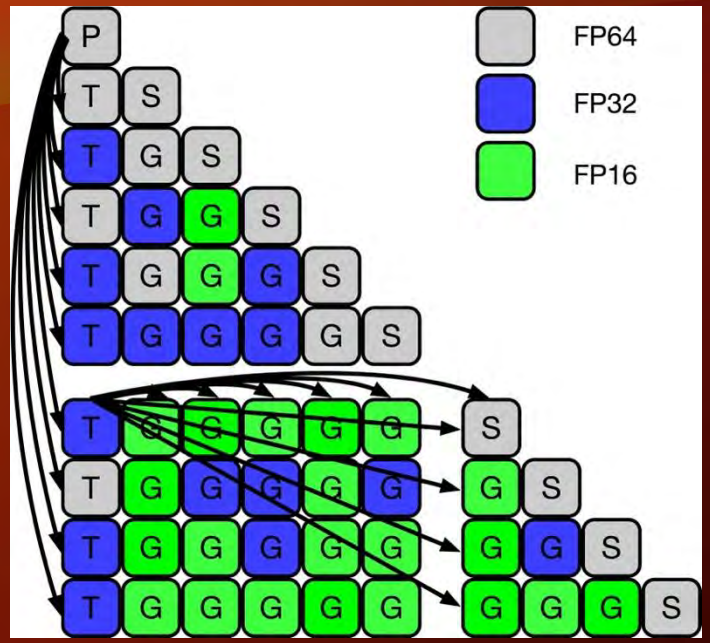
- Mixed-precision algorithms have a long history, e.g., iterative refinement (1963, Wilkinson), where multiple copies of the matrix are kept in different precisions for different purposes.
- There are many such new algorithms; see Higham & Mary, Mixed precision algorithms in numerical linear algebra, Acta Numerica (2022); up to 5 precisions!
- We consider 3 precisions, i.e., double-precision (DP/FP64), single-precision (SP/FP32) and half-precision (HP/FP16), herein and keep just a single matrix copy.

ADAPTIVE MIXED-PRECISION CHOLESKY

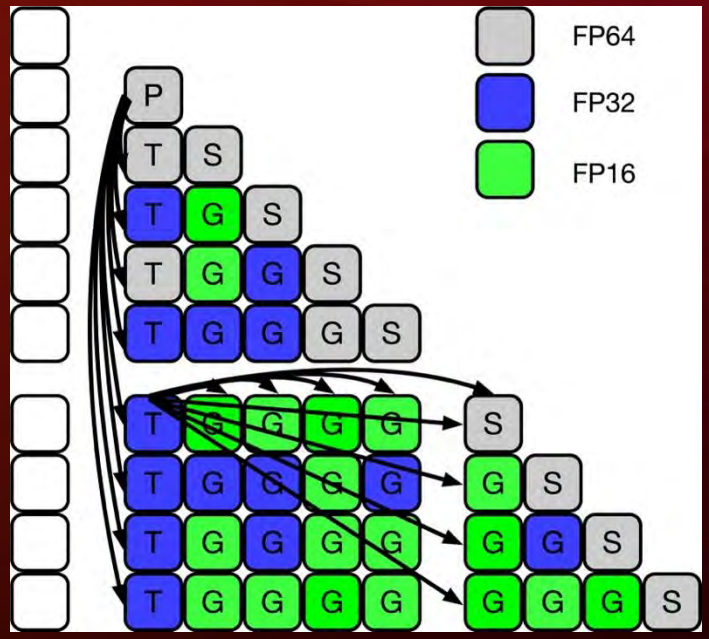


Algorithm 1: Adaptive GPU-based MP Cholesky.

```
1  for k = 0 to NT − 1
2      DPOTRF (k, k, C_kk)
3      for m = k + 1 to NT − 1
4          TRSM (m, k, C*_kk, C_mk)
5          ← DPOTRF (k, k, C*_kk)
6      for m = k + 1 to NT − 1
7          DSYRK (m, k, C*_mk, C_mm)
8          ← TRSM (m, k, C_kk, C*_mk)
9      for m = k + 2 to NT − 1
10         for n = k + 1 to m − 1
11             GEMM (m, n, k, C*_mk, C*_nk, C_mn)
12             ← TRSM (m, k, C_kk, C*_mk)
13             ← TRSM (n, k, C_kk, C*_nk)
```



Panel K = 0

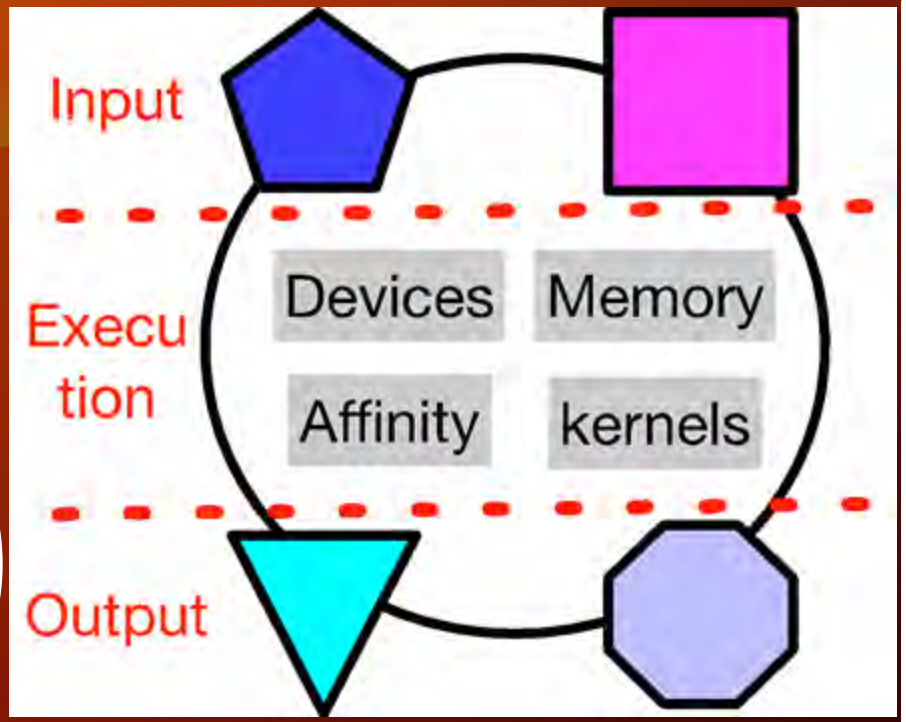


Panel K = 1

Demonstration of the first two iterations in Algorithm 1. P, POTRF; T, TRSM; S, SYRK; G, GEMM. Arrows are representative dependencies introducing communications. The white color indicates numerical kernels on that tile are finished

Challenges: load imbalance, precision conversion, and data movement!!!

HOW PARSEC SOLVES THESE CHALLENGES

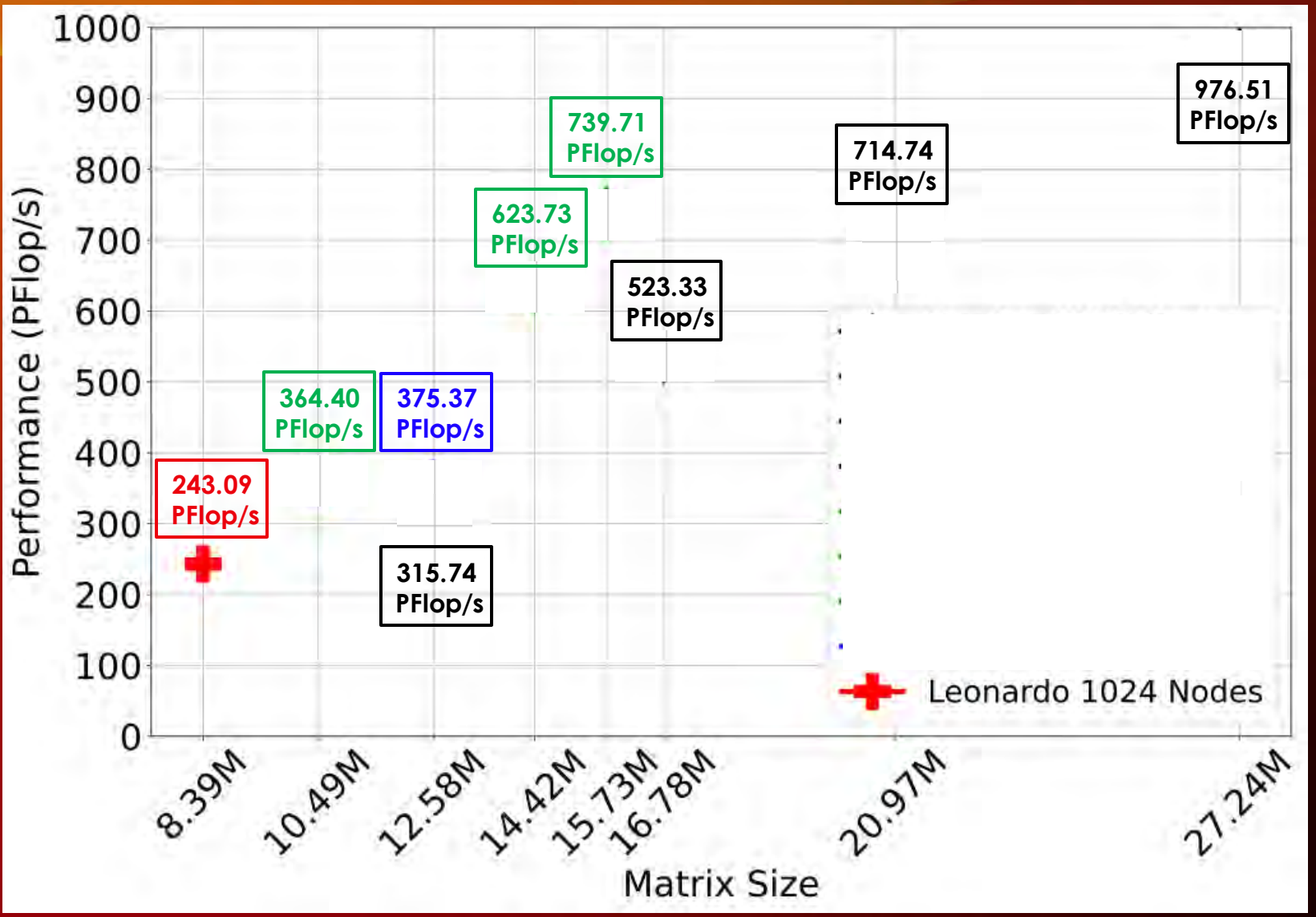


Task's Lifecycle from users' view

- PaRSEC separates (1) data and data distribution, (2) flow and kernel (3) CPU/GPU device (4) memory for each task
- Device-to-device communication and dynamic/user-defined data placement strategies on devices
- Collective communication and dynamic communication volume
- Flows are the same as dense Cholesky; changes are only the data encapsulated in the flow and numerical kernels

Decision made at runtime!!!

PERFORMANCE ON MACHINES OF 4 DIFFERENT GPU FAMILIES



- Performance increases with matrix size **peaking at larger matrix sizes (~27M)**
- **Frontier** achieves **0.975 EFLOP/s** on **9,025 nodes**
- **Summit, Alps, and Leonardo** show a performance range of **~243 to ~739 PFLOP/s**
- **Scaling efficiency improves** with more nodes

SUMMARY

- **Objective:** This work presents the design and implementation of an exascale climate emulator to address the computational and storage challenges of high-resolution Earth System Model (ESM) simulations

- **Methodology:** Uses spherical harmonic transforms for modeling spatio-temporal climate data with tunable resolution, enabling ultra-high spatial resolution emulations (0.034° or ~3.5 km)

- **Data:** The emulator was trained on 318 billion hourly and 31 billion daily temperature data points from global simulation ensembles spanning 35 and 83 years, respectively

SUMMARY

- **Innovations:** Introduces mixed-precision computation strategies optimized for different GPU families, supported by the PaRSEC runtime system, to efficiently balance computation, communication, and memory

- **Performance:** The emulator has been successfully ported to **4 GPU-based systems (Frontier, Alps, Leonardo, and Summit) as well as to the CPU-based system Shaheen III.** It has demonstrated significant computational performance across multiple exascale platforms:
    * 0.976 EFlop/s on 9,025 nodes of Frontier
    * 0.739 EFlop/s on 1,936 nodes of Alps
    * 0.243 EFlop/s on 1,024 nodes of Leonardo
    * 0.375 EFlop/s on 3,072 nodes of Summit

- **Impact:** The emulator enables climate scientists to generate high-resolution climate projections with significantly reduced computational and storage costs, paving the way for more accessible and detailed climate studies

INITIAL TASKS

- Task 0: Read the paper and familiarize with the framework
- Task 0.1: Download the ERA5 dataset two-meter temperature: https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels?tab=overview
- Task 0.2: Read the input, remove the mean trend (stage 1 in this presentation, the code will be provided later)
- Task 0.3: Learn PaRSEC: https://github.com/ICLDisco/parsec/wiki
- Task 1: Install the software (Instruction will be given later)
- More coming soon

THANK YOU!

Paper