Scaling Computational Fluid Dynamics: In Situ Visualization of NekRS using SENSEI

Victor A. Mateevitsi
Argonne National Laboratory
United States of America
vmateevitsi@anl.gov

Mathis Bode
Forschungszentrum Jülich
Jülich Supercomputing Centre
Germany
m.bode@fz-juelich.de

Nicola Ferrier
Argonne National Laboratory
United States of America
nferrier@anl.gov

Paul Fischer
Argonne National Laboratory
University of Illinois
Urbana-Champaign
United States of America
fischerp@illinois.edu

Jens Henrik Göbbert
Forschungszentrum Jülich
Jülich Supercomputing Centre
Germany
j.goebbert@fz-juelich.de

Joseph A. Insley
Argonne National Laboratory
Northern Illinois University
United States of America
insley@anl.gov

Yu-Hsiang Lan
Argonne National Laboratory
University of Illinois
Urbana-Champaign
United States of America
ylan@anl.gov

Misun Min
Argonne National Laboratory
United States of America
mmin@anl.gov

Michael E. Papka
Argonne National Laboratory
University of Illinois at Chicago
United States of America
papka@anl.gov

Saumil Patel
Argonne National Laboratory
United States of America
spatel@anl.gov

Silvio Rizzi
Argonne National Laboratory
United States of America
srizzi@anl.gov

Jonathan Windgassen
Forschungszentrum Jülich
Jülich Supercomputing Centre
Germany
j.windgassen@fz-juelich.de

ABSTRACT

In the realm of Computational Fluid Dynamics (CFD), the demand for memory and computation resources is extreme, necessitating the use of leadership-scale computing platforms for practical domain sizes. This intensive requirement renders traditional checkpointing methods ineffective due to the significant slowdown in simulations while saving state data to disk. As we progress towards exascale and GPU-driven High-Performance Computing (HPC) and confront larger problem sizes, the choice becomes increasingly stark: to compromise data fidelity or to reduce resolution. To navigate this challenge, this study advocates for the use of *in situ* analysis and visualization techniques. These allow more frequent data "snapshots" to be taken directly from memory, thus avoiding the need for disruptive checkpointing. We detail our approach of instrumenting NekRS, a GPU-focused thermal-fluid simulation code employing the spectral element method (SEM), and describe varied *in situ* and in transit strategies for data rendering. Additionally, we provide concrete scientific use-cases and report on runs performed on Polaris, Argonne Leadership Computing Facility's (ALCF) 44 Petaflop supercomputer and Jülich Wizard for European Leadership Science (JUWELS) Booster, Jülich Supercomputing Centre's (JSC) 71 Petaflop High Performance Computing (HPC) system, offering practical insight into the implications of our methodology.

1 INTRODUCTION

Rooted in the Spectral Element Method (SEM) [9], NekRS [8] is a GPU-accelerated thermal-fluid simulation code. Rapidly becoming a staple in modeling and simulating turbulent flows, it finds uses across a wide spectrum, from simulating internal combustion engines to large-scale atmospheric and oceanic flows, as well as reactor thermal hydraulics. The challenges posed by these simulations are significant in terms of scale, resolution, and computational demand. NekRS tackles these hurdles using libParanumal [11, 26],

a high-order finite element solver, and the OCCA library [17, 18], a hardware-agnostic solution, to enable optimized performance and scalability.

With the advent of exascale supercomputers like Argonne National Laboratory's Aurora [23], the disparity between rapid on-chip processing and slower disk storage is set to widen. Data-saving to disk could notably hamper simulations, necessitating pauses for I/O operations to complete. This situation leaves scientists with a tough choice: reduce checkpoint frequency or simplify the domain by lowering the resolution, both potentially resulting in overlooked discoveries.

In situ processing [15], which facilitates data processing while it remains in memory, presents a compelling solution to this challenge. The SENSEI project [4] embodies this approach, aiming to equip the simulation code with the flexibility to interchange *in situ* algorithms without recompilation. In this paper, we demonstrate the instrumentation of NekRS with SENSEI and present our scalability experiments conducted using real scientific use-cases.

2 BACKGROUND

The advancement in computational capabilities has significantly outpaced that of I/O speeds, leading to a heightened interest in *in situ* processing. This technique revolves around the immediate analysis and visualization of data, directly from memory, sidestepping the delays of I/O operations. As a result, simulations become more efficient with reduced cycles spent on intensive I/O. Early adoptions of this method can be traced back to Zajac et al. [25], who showcased a simulation of Earth and satellite orbits plotted straight onto microfilm. However, the rise of heterogeneous systems and the sheer variety of analysis algorithms and graphics Application Programming Interfaces (APIs) like OpenGL, DirectX, Vulkan, and ANARI have compounded the intricacy of embedding universal cross-platform *in situ* capabilities in scientific codes. Dedicated *in situ* libraries are now essential, offering specialized tools for such diverse platforms, simplifying tasks such as image rendering, and easing code instrumentation.

2.1 In situ libraries

Presently, several key libraries, notably Ascent [14], Catalyst [3], LibSim [6, 13], and SENSEI [4] are the most active ones. Ascent, part of the ALPINE project and backed by the U.S. Department of Energy's (DOE) Exascale Computing Project (ECP) [19], is a lightweight *in situ* visualization and analysis library built for multi-physics HPC simulations. Its unique selling point is its minimal reliance on external dependencies, with VTK-m [21] employed for rendering. Catalyst, maintained by Kitware, relies on VTK, facilitating detailed visualization workflows via VTK's comprehensive visualization tools. In a similar vein, LibSim [6, 13] operates alongside VisIt [7].

2.2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) employs numerical methods to study fluid flow problems [1]. Its applications span from designing combustors and simulating nuclear reactors to aerodynamic modeling of aircrafts and space shuttles [12]. Nek5000, pioneered

in the late 80s [9, 10], stands out as a gold standard for spectral element simulations, scaling efficiently across varying computational platforms. Instrumentation studies by Atzori et al. [2] and Bernardoni et al. [5] shed light on its adaptability with *in situ* libraries like Catalyst and SENSEI. Despite Nek5000's enduring prominence over 35 years, the advent of GPU-accelerated HPCs necessitated a major code overhaul, giving rise to its successor, NekRS [8].

3 METHODOLOGY

SENSEI is structured around two primary components: the AnalysisAdaptor and DataAdaptor.

3.1 AnalysisAdaptor

The `AnalysisAdaptor` serves as the interface for *in situ* analysis codes and algorithms, connecting with tools like Catalyst, VTK-m, or OSPray [24]. Developing a new `AnalysisAdaptor` necessitates extending the existing interface and implementing the desired analysis functionality. These adaptors offer flexibility; they can be swapped dynamically at runtime via an `.xml` configuration file. For instance, to enable *in situ* image rendering through Catalyst, one can activate the Catalyst `AnalysisAdaptor` without needing to recompile the entire codebase, as illustrated in Listing 1.

```xml
<sensei>
  <analysis type="catalyst" pipeline="
      pythonscript" filename="analysis.py"
      frequency="100" />
</sensei>
```

Listing 1: SENSEI AnalysisAdaptor Configuration

3.2 DataAdaptor

Data is channeled to the `AnalysisAdaptor` through the use of the `DataAdaptor`. This component is a C++ interface provided by SENSEI, which simulation codes must extend and implement. The role of this child adaptor is to relay data (aligned with the VTK data model) to SENSEI. Moreover, an intermediary "bridge" code is responsible for embedding SENSEI into the simulation, initializing the library, updating the data, managing adaptors, and periodically invoking analysis routines.

```cpp
class DataAdaptor : sensei::DataAdaptor
{
  void Initialize(nek_data) { }
  int GetNumberOfMeshes() { };
  int GetMeshMetadata() { };
  int GetMesh() { };
  int AddArray() { };
};
```

Listing 2: Pseudocode of the DataAdaptor

Given the identical data models employed by both Nek5000 and NekRS, we crafted a unique `nek_sensei::DataAdaptor` class (Listing 2) and corresponding bridge code (Listing 3). To promote reusability and ease maintenance, we housed this code in a separate repository and integrated it into both Nek5000 and NekRS using a GitHub submodule.

NekRS employs OCCA [18], a flexible, vendor-agnostic framework for parallel programming across heterogeneous platforms. When compiled with a GPU backend, such as CUDA, NekRS operates on the GPU. This poses a challenge, as simulation data residing on GPU device memory must be transferred to the CPU before being relayed to SENSEI due to VTK data model's current lack of GPU device memory support.

```
void initialize(MPI_Comm* comm, nek_data) {
  NekDataAdaptor *da = new NekDataAdaptor();
  da->Initialize(nek_data);

  ConfigurableAnalysis ca = new
      ConfigurableAnalysis();
  ca->Initialize("conf.xml");
}

void update(double* t, DataAdaptor **d) { }
```

Listing 3: Pseudocode of the bridge code

For the evaluation, we compiled NekRS latest version (v23), and used SENSEI's development branch. To enable the Catalyst AnalysisAdaptor, we compiled against ParaView 5.11.1, which was compiled with OSPRay support.

4 RESULTS

To evaluate the effectiveness of the instrumentation, we conducted experiments on two distinct HPCs: Polaris, a 44 Petaflops HPE Cray GPU-based HPC, and JUWELS Booster, a 71 Petaflops Atos GPU-based HPC. Polaris houses 560 nodes, each fitted with a single AMD EPYC "Milan" processor and four NVIDIA A100 GPUs. JUWELS Booster consists of 936 compute nodes, each equipped with 2 AMD EPYC "Rome" CPUs and 4 NVIDIA A100 GPUs. The network has a DragonFly+ topology with HDR-200 InfiniBand.

The primary goal of these experiments is to quantify the computational overhead introduced by our workflow. The first use case run on Polaris and is an example for an *in situ* application of our workflow, while the second one run on JUWELS Booster and uses an in transit setup.

To maintain comparability, we show similar performance metrics for both cases. However, we do not perform exactly the same analyses, as the focus is on providing deep insight into the application of different workflow strategies, rather than a one-to-one comparison of the supercomputing environments. For the sake of reproducibility, we have made all source and analysis code, use cases, and data available [16].

4.1 In situ Pebble-bed reactor case

Our assessment hinged on two pivotal metrics: runtime and memory footprint. Herein, runtime refers to the total elapsed wall-clock time, while memory footprint corresponds to the aggregate memory high water mark across all MPI ranks. The configuration adopted for this test include:

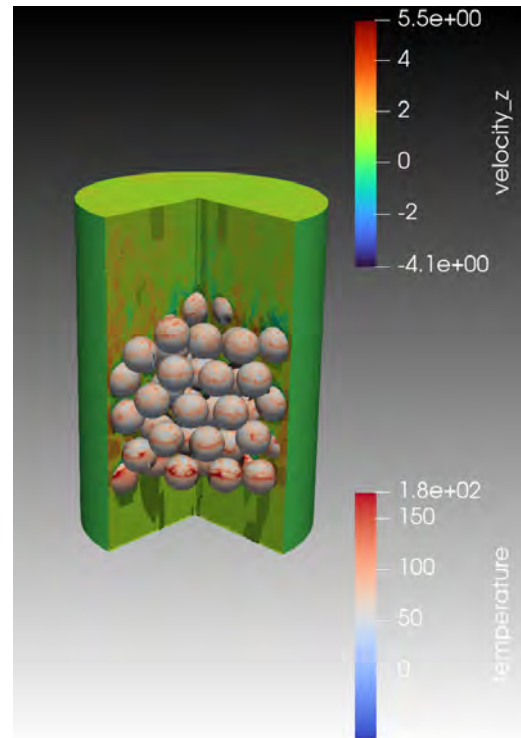- **Original:** Here, NekRS runs sans the SENSEI interface, serving as the baseline.

Figure 1: Visualization of the pb146 use case simulation, illustrating flow dynamics within a pebble-bed nuclear reactor.

- **Checkpointing:** This configuration witnesses NekRS run with built-in checkpointing initiated at every n frames. In this context, checkpointing pertains to the practice of periodically storing raw simulation data onto disk.
- **Catalyst:** In this mode, NekRS, integrated with the SENSEI interface, leverages the Catalyst AnalysisAdaptor. Data is copied from the GPU to the CPU and subsequently passed to SENSEI, which employs the Catalyst Adaptor for rendering tasks.

The test bench used to test our instrumentation was the "pb146" use case simulation, an inherent example within the NekRS suite. This simulation models a computational fluid dynamics representation of a pebble-bed nuclear reactor core, housing 146 spherical pebbles (Figure 1), and runs on the GPUs. Such a simulation is of particular interest, given the growing interest in advanced carbon-neutral nuclear fission reactors [20]. For all configurations, we allowed the simulation to execute for 3,000 timesteps, instigating either checkpointing or *in situ* processes at 100 timestep intervals.

Trials were conducted on 70 nodes (12.5% of Polaris, constituting 280 ranks), 140 nodes (25% of Polaris, which is 560 ranks), and 280 nodes (50% of Polaris, which is 1120 ranks) under both Catalyst and Checkpointing configurations. The elapsed time for the Original configuration was deduced by subtracting the Checkpointing time from the cumulative elapsed time. The outcome is depicted in Figure 2. As expected, the Original configuration showcased optimal time efficiency, unburdened by I/O or *in situ* processing overheads. In
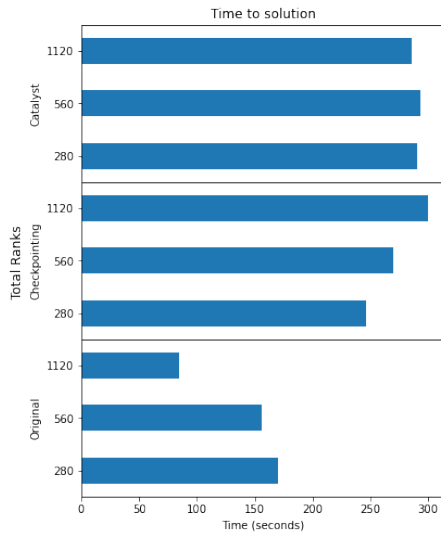
Figure 2: Comparison of time-to-solution across 280, 560 and 1,120 rank runs for Catalyst, Checkpointing, and Original configurations.

Figure 3: Memory usage comparison between the 280, 560 and 1,120 rank runs for both Catalyst and Checkpointing configurations.

Figure 4: Side view visualization of a RBC case.

juxtaposition, the Catalyst approach bore a slight overhead when pitted against Checkpointing. However, it's crucial to highlight that the storage demand for Catalyst was a mere 6.5MB, in stark contrast to the whopping 19GB necessitated by Checkpointing. This signifies that, while the computational overheads of *in situ* almost mirror those of Checkpointing, they accomplish this at an impressive storage economy, nearly three orders of magnitude less. Delving deeper, Figure 3 illustrates that Catalyst's CPU memory overhead is approximately 25% greater. This escalation is rational, given the need to transition data from GPU to CPU and the inherent overhead accompanying Catalyst operations.

4.2 In transit Mesoscale case

Rayleigh-Bénard convection (RBC) is one of the classical natural convection types in fluid thermodynamics and has been widely studied (Fig. 4). A basic setup leading to RBC is a fluid heated from below (i.e., perpendicular to the direction of gravity). Depending on the so-called Rayleigh number (Ra), the heat transfer in such a setup is dominated either by conduction (low Ra) or convection (high Ra), and a so-called Bénard cell is formed.

Numerically, high Ra setups are very interesting because these flows are highly turbulent, leading to a large separation of scales. Therefore, large meshes with high resolution are required and simulations are only possible with supercomputers, making it a perfect example for our workflow. A recent example of RBC computed on supercomputers used parameters relevant to conditions at the Sun's surface, resulting in so-called mesoscale convection [22].

For our scaling measurements, NekRS-SENSEI is complemented by ADIOS2 (v.2.9.1) for data transport, resulting in an in transit visualization workflow. A major advantage of the in-transit workflow is that the memory available for simulation nodes is independent of the number of visualization cores/nodes. Since available memory is
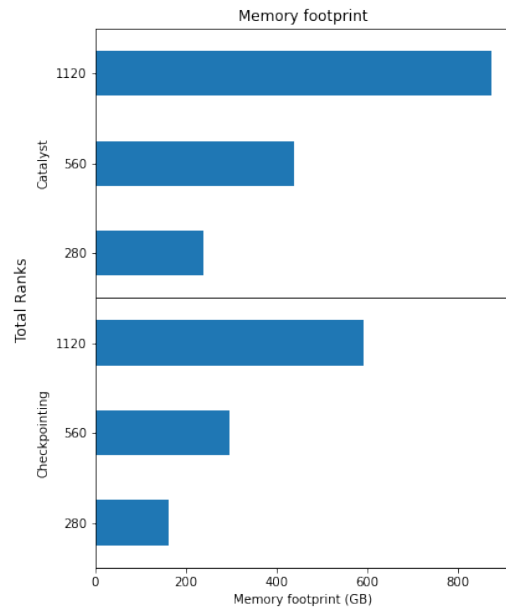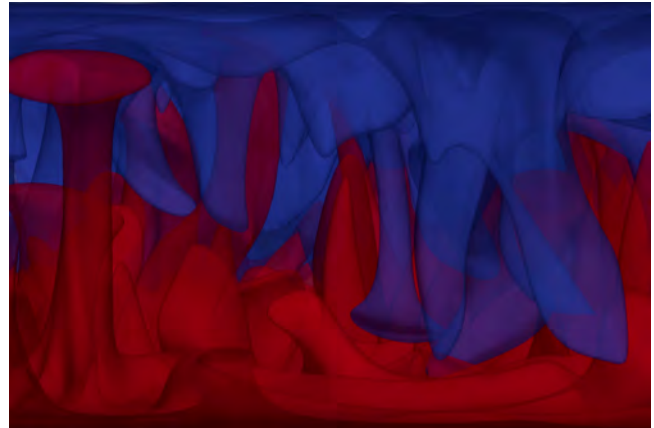
often one of the limiting parameters for simulation size, this avoids unnecessary trade-offs between simulation size and visualization speed. The endpoint of our workflow is always a SENSEI data consumer, and the ratio of simulation nodes to endpoint nodes is 4:1 in all cases.

The Sustainable Staging Transport (SST) engine with its classic streaming data architecture is selected as the ADIOS2 engine. It is configured to communicate via UCX for data transport and is set to use TCP sockets on Infiniband for control operations and BP as a data marshaling option.

This in-transit workflow is evaluated using three measurement points:

- **No Transport:** For this reference measurement, no SENSEI analysis adapter is enabled in the SENSEI runtime XML configuration. However, SENSEI is still used for the measurement.
- **Checkpointing:** The SENSEI endpoint is configured to write the pressure and velocity fields to the storage system as VTU files.
- **Catalyst:** The SENSEI endpoint receives the data from NekRS-SENSEI and renders two images using ParaView over Python.

As before for the *in situ* case, our analysis for the in transit case focuses on the overhead in terms of time (Fig. 5) and memory (Fig. 6) caused by the visualization for the simulation node. Furthermore, the scalability is evaluated by means of weak scaling, i.e. the theoretical load per node is kept constant as the number of nodes is increased.
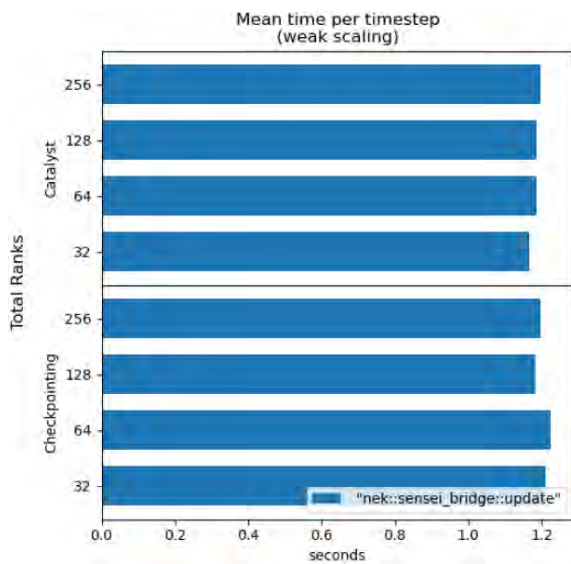

Figure 5: Measurement of the mean time per timestep on the NekRS-SENSEI simulation nodes. Each rank represents one GPU.

The measurements shown in Figure 5 highlight two important points for the in transit workflow. First, the times for the Catalyst and Checkpointing measurement points are very similar, meaning that the in transit overhead is small. Second, the times for increasing nodes do not increase significantly. Therefore, weak scaling works well, and it can be assumed that the workflow will work for even larger setups.

This result is also supported by Figure 6. The memory consumption for Catalyst and No Transport is very similar. The memory overhead of Checkpointing is visible, but not very large. Again, note that the memory available for the simulation nodes is independent of the number of visualizers.

5 DISCUSSION AND CONCLUSION

Across both the *in situ* Pebble-bed reactor case and the *in transit* Mesoscale case, we have evaluated the computational paradigms
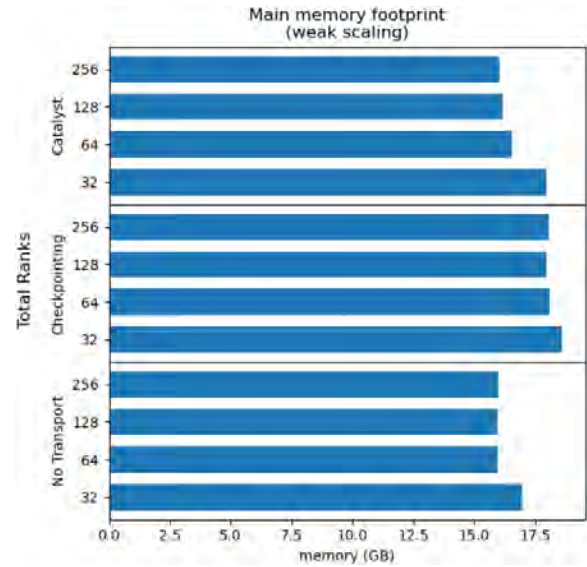

Figure 6: Measurement of the main memory footprint per NekRS-SENSEI simulation node. Each rank represents one GPU.

that leverage contemporary *in situ* and in transit visualization methodologies. This exploration aimed to balance computational efficiency, storage necessities, and effective visualization in the dynamic landscape of high-performance computing.

A recurring theme in our assessments is the value that advanced visualization brings to computational workloads. Whether analyzing the flow dynamics within a pebble-bed reactor or exploring the turbulent flows of Rayleigh-Bénard convection, the ability to "see" the data in real-time significantly augments our analytical capacities. Both case studies underscore the importance of managing computational efficiency in conjunction with data storage and visualization requirements. The Catalyst approach in the Pebble-bed reactor case presented an impressive storage economy, demonstrating that efficient visualization doesn't need to come at the expense of increased storage demands. Similarly, the in-transit approach for the Mesoscale case showed that with the right tools and configuration, overheads can be minimized, preserving the sanctity of computational resources. One of the key markers of success for any high-performance computing methodology is scalability. Our results, especially in the context of the in-transit visualization of Rayleigh-Bénard convection, attest to the fact that these approaches are not merely academic exercises but are scalable, efficient, and ready for the rigors of future complex simulations.

In conclusion, as the gap between I/O and computational demands widens, methodologies like *in situ* and in transit analysis and visualization offer promising pathways. Our findings demonstrate that with careful configuration, integration of advanced tools, and an understanding of the underlying phenomena, we can achieve computational efficiency without compromising on visualization efficacy. As computational simulations become even more intricate and demand increased resources, such approaches will be pivotal in advancing scientific understanding.

ACKNOWLEDGMENTS

REFERENCES

[1] John David Anderson and John Wendt. 1995. *Computational fluid dynamics*. Vol. 206. Springer.

[2] Marco Atzori, Wiebke Köpp, Steven W. D. Chien, Daniele Massaro, Fermín Mallor, Adam Peplinski, Mohamad Rezaei, Niclas Jansson, Stefano Markidis, Ricardo Vinuesa, Erwin Laure, Philipp Schlatter, and Tino Weinkauf. 2022. In situ visualization of large-scale turbulence simulations in Nek5000 with ParaView Catalyst. *The Journal of Supercomputing* 78, 3 (Feb. 2022), 3605–3620. https://doi.org/10.1007/s11227-021-03990-3

[3] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV2015)*. Association for Computing Machinery, New York, NY, USA, 25–29. https://doi.org/10.1145/2828612.2828624

[4] Utkarsh Ayachit, Brad Whitlock, Matthew Wolf, Burlen Loring, Berk Geveci, David Lonie, and E. Wes Bethel. 2016. The SENSEI Generic In Situ Interface. In *2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. 40–44. https://doi.org/10.1109/ISAV.2016.013

[5] Bennett Bernardoni, Nicola Ferrier, Joseph Insley, Michael E Papka, Saumil Patel, and Silvio Rizzi. 2018. In situ visualization and analysis to design large scale experiments in computational fluid dynamics. In *2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 94–95.

[6] Hank Childs. 2012. In Situ Processing. (Nov. 2012). https://escholarship.org/uc/item/3st8x19d

[7] Hank Childs. 2012. VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data. (Nov. 2012). https://escholarship.org/uc/item/69r5m58v

[8] Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, and Tim Warburton. 2022. NekRS, a GPU-accelerated spectral element Navier–Stokes solver. *Parallel Comput.* 114 (Dec. 2022), 102982. https://doi.org/10.1016/j.parco.2022.102982

[9] Paul Fischer, Einar M. Ronquist, Daniel Dewey, and Anthony T. Patera. 1988. *Spectral element methods: Algorithms and architectures*. Technical Report NAS 1.26:182701. https://ntrs.nasa.gov/citations/19880011494 NTRS Author Affiliations: Massachusetts Inst. of Tech. NTRS Document ID: 19880011494 NTRS Research Center: Legacy CDMS (CDMS).

[10] Paul Frederick Fischer. 1989. *Spectral element solution of the Navier-Stokes equations on high performance distributed-memory parallel processors*. PhD Thesis. Massachusetts Institute of Technology.

[11] A. Karakus, N. Chalmers, K. Świrydowicz, and T. Warburton. 2019. A GPU accelerated discontinuous Galerkin incompressible flow solver. *J. Comput. Phys.* 390 (Aug. 2019), 380–404. https://doi.org/10.1016/j.jcp.2019.04.010

[12] Essam E. Khalil. 2021. CFD History and Applications. *ARCHIVES OF AKADEMIA BARU ARTICLES* 4, 2 (July 2021), 43–46. https://www.akademiabaru.com/index.php/archives/article/view/278 Number: 2.

[13] T Kuhlen, R Pajarola, and K Zhou. 2011. Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*, Vol. 10. Eurographics Association Aire-la-Ville, Switzerland, 101–109.

[14] Matthew Larsen, Eric Brugger, Hank Childs, and Cyrus Harrison. 2022. Ascent: A Flyweight In Situ Library for Exascale Simulations. In *In Situ Visualization for Computational Science (Mathematics and Visualization)*, Hank Childs, Janine C. Bennett, and Christoph Garth (Eds.). Springer International Publishing, Cham, 255–279. https://doi.org/10.1007/978-3-030-81627-8_12

[15] Kwan-Liu Ma, Chaoli Wang, Hongfeng Yu, and Anna Tikhonova. 2007. In-situ processing and visualization for ultrascale simulations. *Journal of Physics: Conference Series* 78, 1 (July 2007), 012043. https://doi.org/10.1088/1742-6596/78/1/012043

[16] Victor A. Mateevitsi, Mathis Bode, Nicola Ferrier, Paul Fischer, Jens Henrik Göbbert, Joseph A. Insley, Yu-Hsiang Lan, Misun Min, Michael E. Papka, Saumil Patel, Silvio Rizzi, and Jonathan Windgassen. 2023. Software and Analysis for paper: Scaling Computational Fluid Dynamics: In Situ Visualization of NekRS using SENSEI. https://doi.org/10.5281/zenodo.8377974

[17] David Medina. 2015. OKL: A Unified Language for Parallel Architectures. (June 2015). https://scholarship.rice.edu/handle/1911/102233 Accepted: 2018-06-19T17:49:54Z.

[18] David S. Medina, Amik St-Cyr, and T. Warburton. 2014. OCCA: A unified approach to multi-threading languages. https://doi.org/10.48550/arXiv.1403.0968 arXiv:1403.0968 [cs].

[19] Paul Messina. 2017. The Exascale Computing Project. *Computing in Science & Engineering* 19, 3 (May 2017), 63–67. https://doi.org/10.1109/MCSE.2017.57 Conference Name: Computing in Science & Engineering.

[20] Misun Min, Yu-Hsiang Lan, Paul Fischer, Elia Merzari, Stefan Kerkemeier, Malachi Phillips, Thilina Rathnayake, April Novak, Derek Gaston, Noel Chalmers, and Tim Warburton. 2022. Optimization of full-core reactor simulations on summit. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '22)*. IEEE Press, Dallas, Texas, 1–11.

[21] Kenneth Moreland, Christopher Sewell, William Usher, Li-ta Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, Matthew Larsen, Chun-Ming Chen, Robert Maynard, and Berk Geveci. 2016. VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications* 36, 3 (May 2016), 48–58. https://doi.org/10.1109/MCG.2016.48 Conference Name: IEEE Computer Graphics and Applications.

[22] Ambrish Pandey, Dmitry Krasnov, Katepalli R Sreenivasan, and Jörg Schumacher. 2022. Convective mesoscale turbulence at very low Prandtl numbers. *Journal of Fluid Mechanics* 948 (2022), A23. Publisher: Cambridge University Press.

[23] Rick Stevens, Jini Ramprakash, Paul Messina, Michael Papka, and Katherine Riley. 2019. *Aurora: Argonne's Next-Generation Exascale Supercomputer*. Technical Report. Argonne National Lab. (ANL), Argonne, IL (United States). https://www.osti.gov/sciencecinema/biblio/1562918

[24] I Wald, GP Johnson, J Amstutz, C Brownlee, A Knoll, J Jeffers, J Günther, and P Navratil. 2017. OSPRay - A CPU Ray Tracing Framework for Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 931–940. https://doi.org/10.1109/TVCG.2016.2599041 Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[25] E. E. Zajac. 1964. Computer-made perspective movies as a scientific and communication tool. *Commun. ACM* 7, 3 (March 1964), 169–170. https://doi.org/10.1145/363958.363993

[26] Kasia Świrydowicz, Noel Chalmers, Ali Karakus, and Tim Warburton. 2019. Acceleration of tensor-product operations for high-order finite element methods. *The International Journal of High Performance Computing Applications* 33, 4 (July 2019), 735–757. https://doi.org/10.1177/1094342018816368 Publisher: SAGE Publications Ltd STM.