

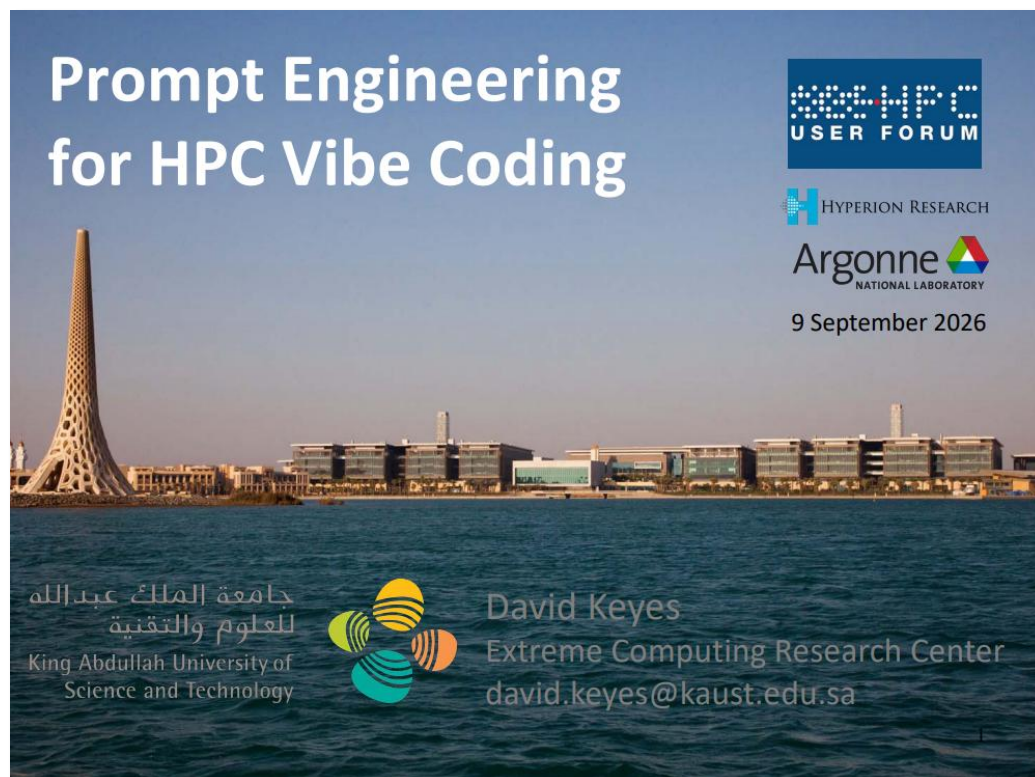
HPC User Forum Update

Prompt Engineering for HPC Vibe Coding

Mark Nossokoff
October 2026

IN THIS UPDATE

The HPC User Forum was established in 1999 to promote the health of the global HPC industry and address issues of common concern to users. In September 2026, the 92nd HPC User Forum took place at Argonne National Lab in Lemont, IL. This update summarizes a presentation from that conference given by David Keyes, Professor, Applied Mathematics and Computational Science and Principal Investigator, Hierarchical Computations on Manycore Architectures at the Extreme Computing Research Center at King Abdullah University of Science and Technology (KAUST) in Thuwal, Saudi Arabia. Keyes explored how AI-assisted coding can be guided by proven principles from high performance computing and positioned the talk as a bridge between traditional computational science and emerging AI coding tools. His goal was to show that effective prompts should reflect deep understanding of algorithms and computer architecture.



Source: Keyes, 2026

PRESENTATION: PROMPT ENGINEERING FOR HPC VIBE CODING, DAVID KEYES, PROFESSOR, APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE, AND PRINCIPAL INVESTIGATOR, HIERARCHICAL COMPUTATIONS

Keyes opened by suggesting that the principles that made traditional high performance computing successful are becoming even more important in the age of AI-assisted software development. While generative AI can help create code more quickly, organizations still need a deep understanding of computer architecture, algorithms, data movement, energy efficiency, and scalability. He proceeded to outline 18 imperatives with examples to help developers create faster, more scalable, and more energy-efficient applications:

1. Reside High on the Memory Hierarchy
2. Reduce Message Number and Volume
3. Reduce Frequency of Synchronization
4. Reduce Span of Synchronization
5. Embrace Heterogeneity
6. Exploit High-Order Discretizations
7. Exploit Hierarchical Representations
8. Exploit Data Sparsity
9. Exploit Nested Levels of Parallelism
10. Exercise the Right to Reorder
11. Employ Dynamic Scheduling and Balancing
12. Co-design Algorithms with Hardware
13. Take Resilience into Algorithms
14. Process on the Fly
15. Code to Specialized Backends
16. Avoid Over-Resolving and Over-Solving
17. Maximize Science per Joule
18. Reformulate Before Computing

The Imperatives

1. Reside High on the Memory Hierarchy

Modern processors improve much faster than memory systems. As a result, moving data often costs more time and energy than performing calculations. HPC applications must keep data as close to the processor as possible to avoid memory bottlenecks. The goal is to reduce data movement and increase cache reuse. Developers can use hierarchical data structures and compression techniques that fit naturally within modern memory hierarchies.

The presentation highlights hierarchical and low-rank matrix methods that reduce storage requirements dramatically. Smaller data footprints allow larger problems to remain in fast memory and improve overall performance. Memory efficiency is now a primary performance driver. Organizations that optimize memory first may gain larger benefits than those focusing only on raw compute power.

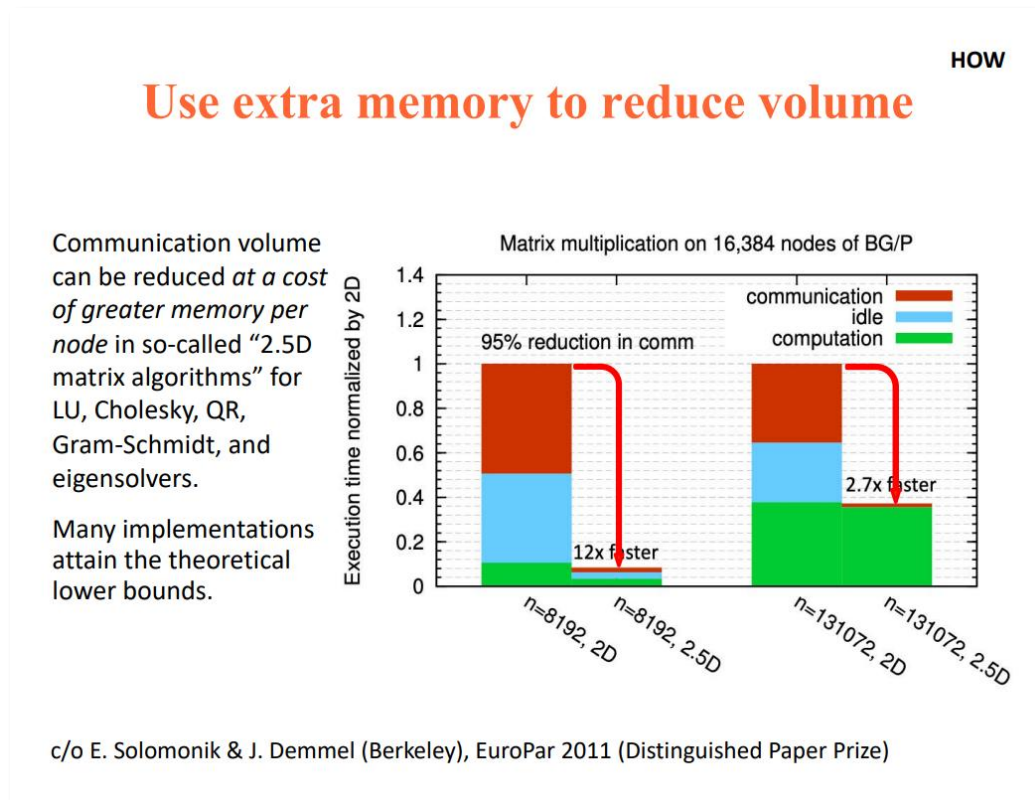
2. Reduce Message Number and Volume

Communication between nodes is becoming relatively more expensive than computation. Large messages and frequent data exchanges can limit scalability. Applications should minimize the amount of data transferred across the network. This may require using more memory or changing algorithms.

Keyes provided an example of 2.5D matrix algorithms, which store extra copies of data to reduce network traffic. Many implementations approach theoretical communication limits. The tradeoff shows that spending more memory can save significant execution time in large systems.

FIGURE 1

Using Extra Memory to Reduce Volume



Source: Keyes, 2026

3. Reduce Frequency of Synchronization

Global synchronization forces processors to wait for one another. As systems scale, these waiting periods become increasingly costly. A common strategy is to perform extra local computation before synchronizing. This reduces how often global coordination is required.

The presentation included an example of communication-avoiding conjugate gradient methods. Bundling reductions decreases synchronization points while maintaining accuracy.

4. Reduce Span of Synchronization

Not all synchronization costs come from frequency. The number of participants involved in a synchronization event also matters. Algorithms can reduce the scope of coordination by solving parts of a problem locally before interacting globally.

An example of nonlinear Schwarz and related preconditioning approaches was shared. Local solves reduce the number of expensive global iterations required.

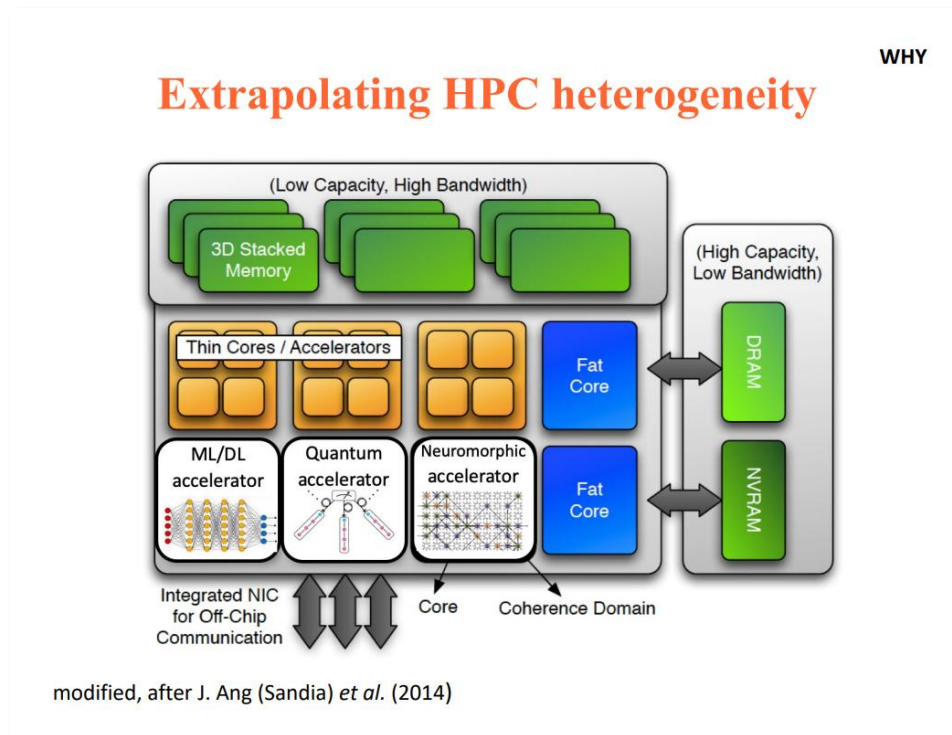
5. Embrace Heterogeneity

Modern supercomputers increasingly combine CPUs, GPUs, and potentially quantum processors. Homogeneous systems are becoming less common. Applications should use each processor type for the tasks it performs best. Resource-aware scheduling becomes a key design requirement.

The presentation highlights variational quantum eigensolvers, where quantum and classical resources work together. Each processor contributes its strengths to the workflow. Future AI and HPC systems will likely depend on effective coordination across diverse hardware types.

FIGURE 2

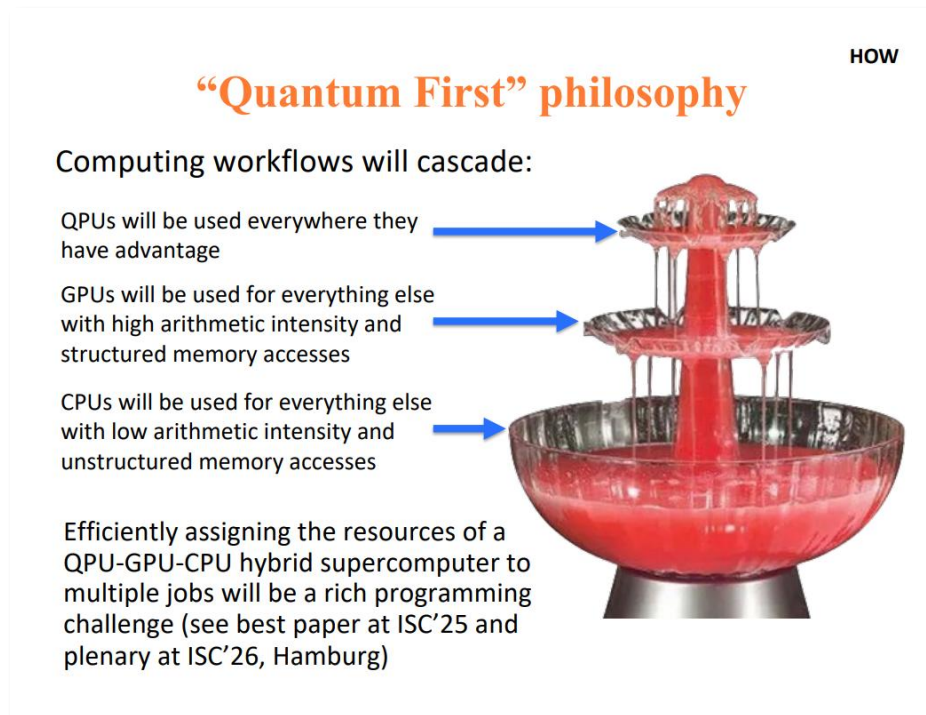
Extrapolating HPC heterogeneity



Source: Keyes, 2026

FIGURE 3

"Quantum First" Philosophy



Source: Keyes, 2026

6. Exploit High-Order Discretizations

Higher-order numerical methods can deliver greater accuracy with fewer degrees of freedom. This improves arithmetic intensity and efficiency. Rather than using many low-order elements, developers can employ fewer high-order elements. This increases useful computation per data item.

Keyes shared an example that showed spectral element methods achieving similar accuracy with substantially lower runtime. The approach delivers both better performance and reduced memory requirements.

7. Exploit Hierarchical Representations

Algorithmic improvements often outperform hardware improvements over the long term. Hierarchical methods reduce computational complexity. Many successful algorithms organize data recursively. This allows large problems to be represented and solved more efficiently.

Examples include multigrid, fast multipole methods, hierarchical matrices, and tensor techniques. The key lesson is that changing the mathematical representation can produce transformational performance gains.

8, Exploit Data Sparsity

Many scientific datasets contain patterns that can be compressed without losing important information. Low-rank and sparse techniques reduce memory use and computation.

The presentation discussed tile low-rank decompositions for seismic processing on Cerebras hardware. Compression enabled larger problems to fit within available memory. Recognizing exploitable structure is often the difference between a feasible and infeasible computation.

9. Exploit Nested Levels of Parallelism

Modern systems provide parallelism at many levels, from threads to nodes to entire workflow ensembles. Applications should leverage all available layers of concurrency.

Keyes shared an example that wrapped swarm optimization around a parallel numerical kernel. Multiple independent searches execute simultaneously. Combining forms of parallelism maximizes resource utilization and throughput.

10, Exercise the Right to Reorder

The order of operations can greatly affect memory access patterns and execution efficiency. Developers can reorganize data and computations to improve locality.

The presentation cited mesh reordering, blocking, and space-filling curve approaches. These methods improve cache behavior and reduce access costs. Thoughtful reordering can provide significant speedups without changing scientific results.

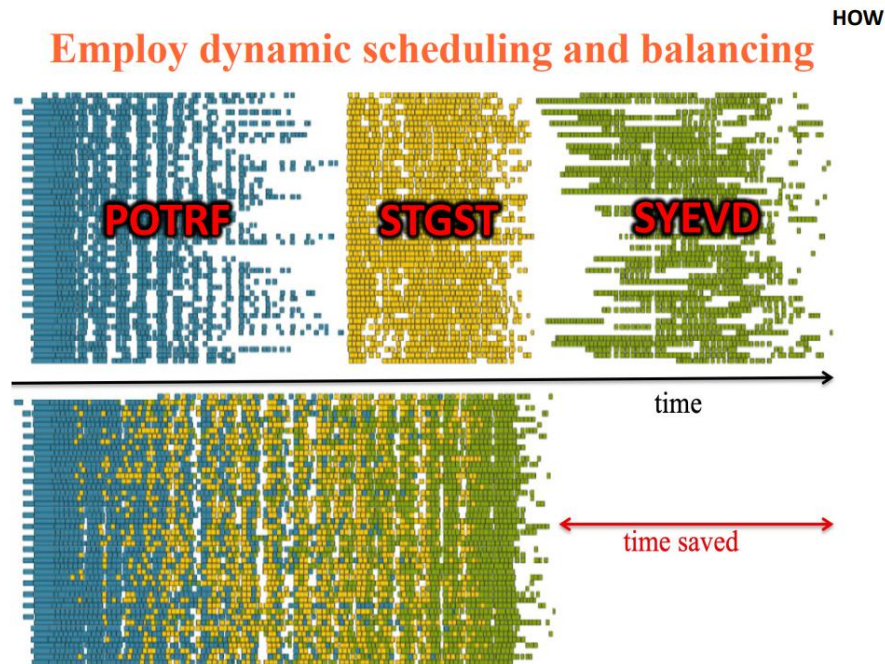
11. Employ Dynamic Scheduling and Balancing

Static schedules often leave resources idle when workloads become uneven. Dynamic runtime systems can assign work as resources become available.

The presentation included an example that showed how tile algorithms benefiting from runtime scheduling instead of rigid execution order. Better load balancing increases throughput and reduces wasted computing capacity.

FIGURE 4

Employ Dynamic Scheduling and Balancing



Source: Keyes, 2026

12. Co-design Algorithms with Hardware

Hardware capabilities and algorithms should evolve together instead of independently. Developers should understand platform features and incorporate them directly into algorithm design.

The FFT example shared used communication capabilities built into MPI and networks to perform transposes efficiently. Co-design often unlocks performance that traditional software approaches cannot reach.

13. Take Resilience into Algorithms

Large systems experience failures, errors, and potentially malicious behavior. Relying entirely on hardware reliability is expensive. Algorithms can become more fault tolerant through statistical and mathematical techniques.

The presentation discussed variance reduction methods that maintain learning progress despite Byzantine attacks. Future exascale and distributed AI systems will likely require resilience as a design feature.

14. Process on the Fly

Storing intermediate results can consume large amounts of memory and bandwidth. Applications should compute values when needed and avoid unnecessary data movement.

Matrix-free finite-element pipelines provide was shared as an example. Operations are fused and executed without assembling large matrices.vReducing storage can improve both performance and energy efficiency.

15. Code to Specialized Backends

Accelerators contain highly optimized execution units that deliver exceptional performance. Software should provide a common interface while targeting specialized hardware underneath.

The presentation emphasized SIMD, SIMT, tensor cores, and AI-oriented hardware features. Organizations gain portability for users while still capturing vendor-specific performance.

FIGURE 5

NVIDIA Roadmap Improvements

WHY

**Tracking NVIDIA's
superlinear-in-inverse-Byte-ratio
improvements (through Blackwell)**

Peak Performance in TF/s	V100 NVLink	A100 NVLink	H100 SXM	B200
FP64	7.5	9.7	34	90
FP32		19.5	67	40
FP64 Tensor Core	15	19.5	67	40
FP/TF32 Tensor Core	8x	156	495	1125
FP16 Tensor Core	120	312	990	2250 225x
FP8/INT8 Tensor Core	-	624	1980	4500
FP4 Tensor Core	-	-	-	9000

Source: Keyes, 2026

16. Avoid Over-Resolving and Over-Solving

Many simulations solve problems more accurately than required. This wastes time and energy. Accuracy targets should be tied directly to the desired outputs. Resolution and tolerances can increase only when needed.

The example provided in the presentation focused on engineering outputs such as heat flux, where uniform high accuracy everywhere is unnecessary. Smarter accuracy management reduces cost without sacrificing decision quality.

17. Maximize Science per Joule

Energy consumption is now a core measure of HPC success. Algorithms should balance speed, precision, and resource usage to reduce power demands.

The presentation showed mixed-precision and low-rank methods lowering both runtime and energy consumption. The future of HPC may be shaped as much by sustainability goals as by performance goals.

18. Reformulate Before Computing

Sometimes the best optimization is to change the problem itself. Reformulation can transform a difficult computation into one that matches high-performance hardware.

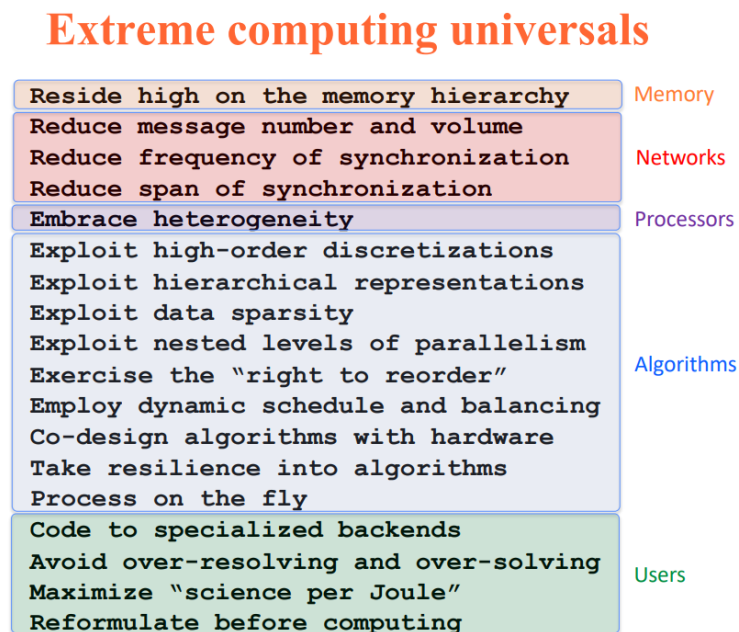
The Hamming distance example converted vector operations into matrix operations that run efficiently on CPUs and GPUs. This imperative captured the presentation's central message: think carefully before coding, even when using AI-assisted development tools.

Conclusion

Keyes concluded the presentation showing how multiple elements of system architecture are involved, along with the oftentimes overlooked human element.

FIGURE 6

Tying it All Together



Source: Keyes, 2026

For more information or to view this and other presentations given at HPC User Forums dating back to 2008, visit www.hpcuserforum.com.

About Hyperion Research, LLC

Hyperion Research provides data-driven research, analysis and recommendations for technologies, applications, and markets in high performance computing and emerging technology areas to help organizations worldwide make effective decisions and seize growth opportunities. Research includes market sizing and forecasting, share tracking, segmentation, technology, and related trend analysis, and both user and vendor analysis for multi-user technical server technology used for HPC, AI, quantum computing, and cloud computing. Hyperion Research provides thought leadership and practical guidance for users, vendors, and other members of the advanced computing community by focusing on key market and technology trends across government, industry, commerce, and academia.

Headquarters

365 Summit Avenue
St. Paul, MN 55102
USA
612.812.5798

www.HyperionResearch.com and www.hpcuserforum.com

Copyright Notice

Copyright 2026 Hyperion Research LLC. Reproduction is forbidden unless authorized. All rights reserved. Visit www.HyperionResearch.com to learn more. Please contact 612.812.5798 and/or email info@hyperionres.com for information on reprints, additional copies, web rights, or quoting permission.