

Architectural requirements for a hybrid GPU/CPU middleware

Burkhard Zink

Center for Computation and Technology (LSU)

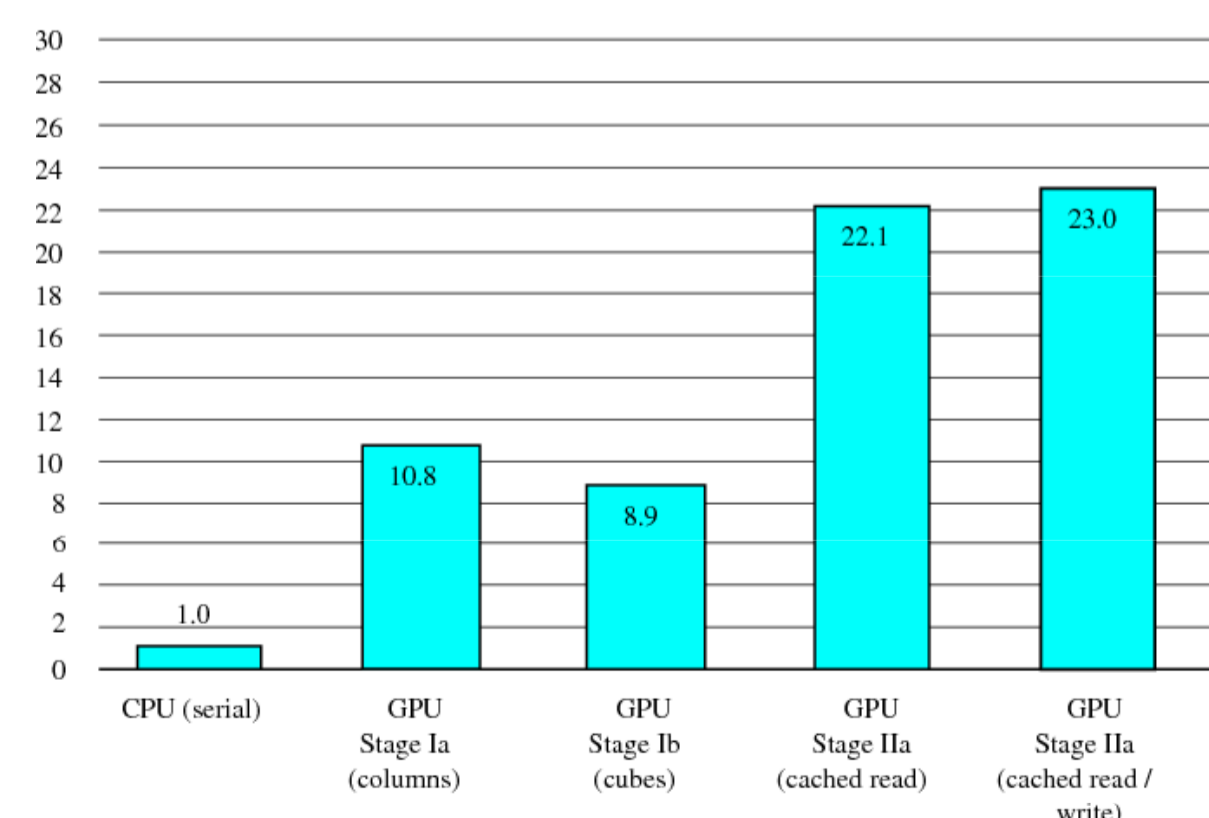
Abstract

Graphics Processing Units (GPUs) can speed up scientific codes by significant amounts. However, it can be challenging to implement ports on a GPU for application scientists. Middlewares like CACTUS are well suited to deliver the needed abstractions. I describe architectural requirements to enable hybrid GPU/CPU computing in middleware solutions.

Performance through GPUs

A single NVIDIA G80 GPU currently delivers up to 350 GFlop/s for about \$1,000. Though still limited to single-precision operations, new hardware supporting double-precision operations has been announced by AMD and NVIDIA.

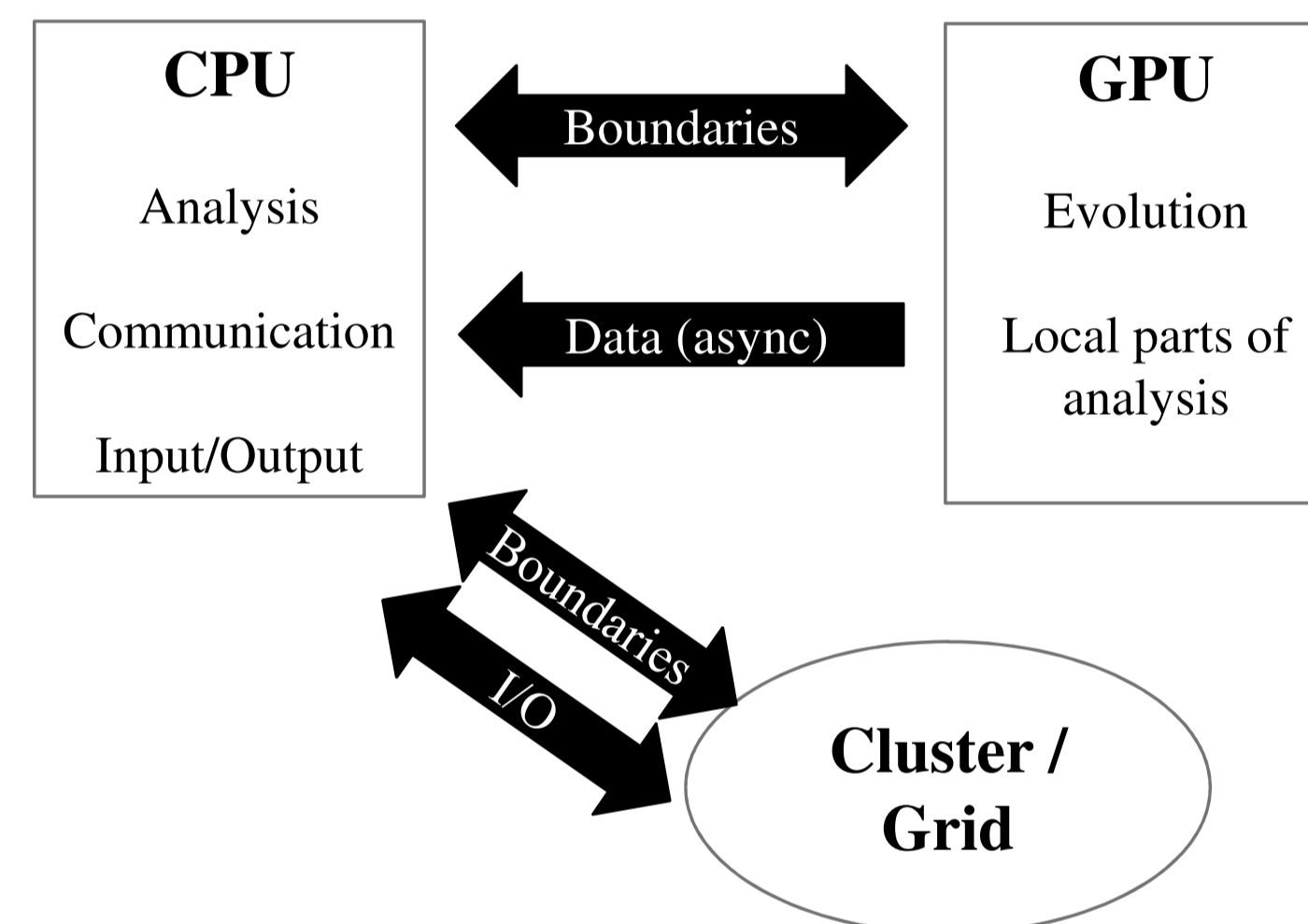
Can scientific codes be accelerated efficiently by a GPU? The CCT has performed prototype experiments with a code used in astrophysics to simulate gravitational fields. This needs to be done, traditionally, on large parallel machines.



Performance enhancement from using a GPU for simulating gravitational fields. The serial CPU code runs on an AMD Opteron 2.4 GHz. Different GPU ports are shown, with speed-ups of up to 23.

Hybrid GPU/CPU computing

GPUs are particularly well-suited to compute highly parallel problems, e.g. the calculation of a right-hand side in a finite-difference problem. At the same time, the CPU can handle tasks like input/output, analysis, and inter-node communication asynchronously.



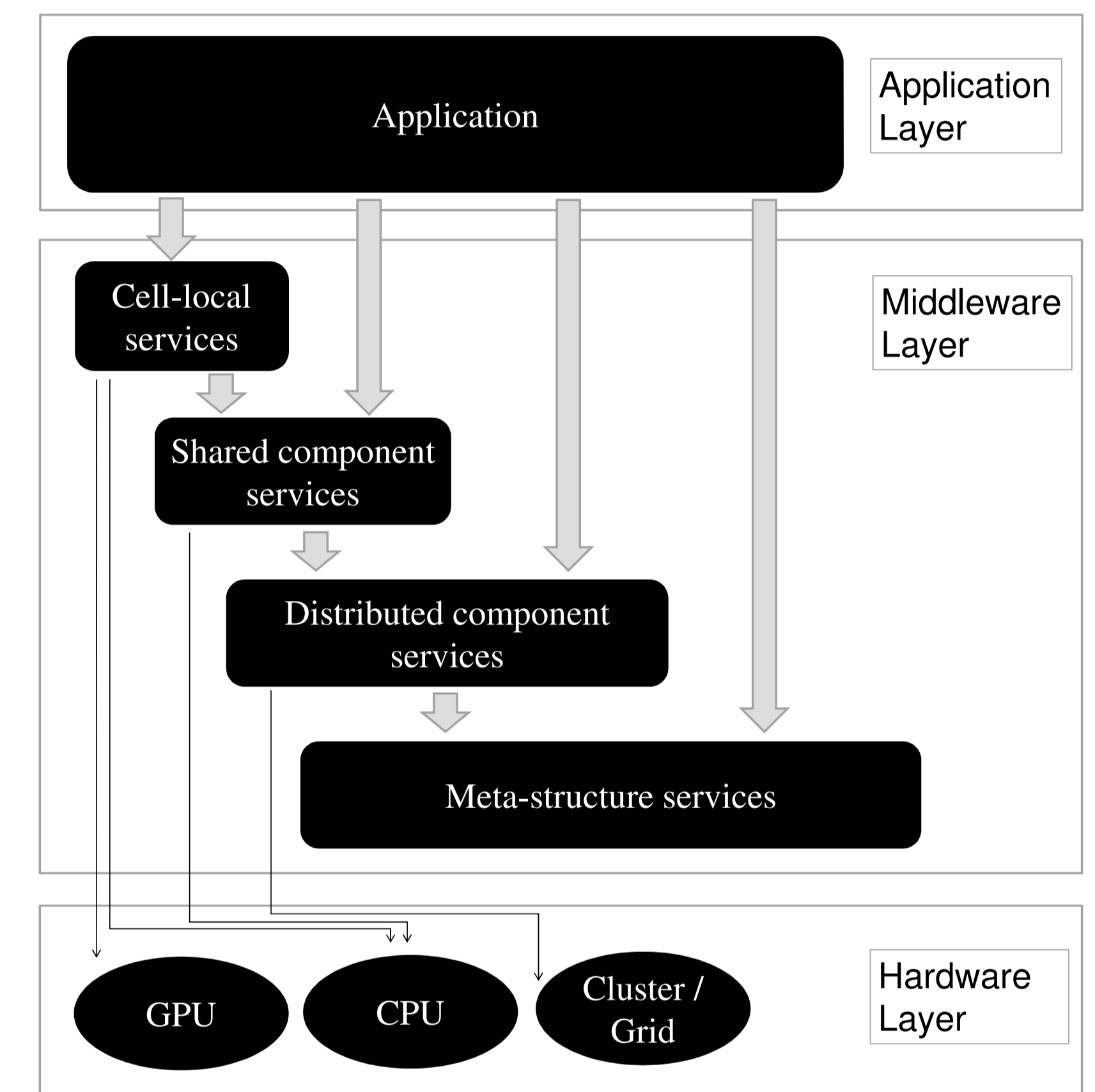
Requirements for middleware

A middleware must support sufficient levels of abstraction to easily switch between pure CPU and hybrid environments, whether on single workstations, parallel clusters, or distributed grids.

The main quality attributes are:

- **Portability.** The introduction of new GPU architectures and programming paradigms should not require changes in the high-level scientific code.
- **Usability.** The application scientist must be presented with an easy-to-use software environment and interfaces.
- **Flexibility.** At the same time, low-level interfaces should be available which enable the implementation of hand-tailored algorithms if needed.

Base architecture



For fully hybrid portable code, the application uses only cell-local services of the middleware, which provides input/output and distributed data-structures by means of its lower level service layers. In turn, these services are also directly visible to the application scientist for maximal flexibility.

References

Burkhard Zink: *A general relativistic evolution code on CUDA architectures*, CCT Technical Report Series (2008)