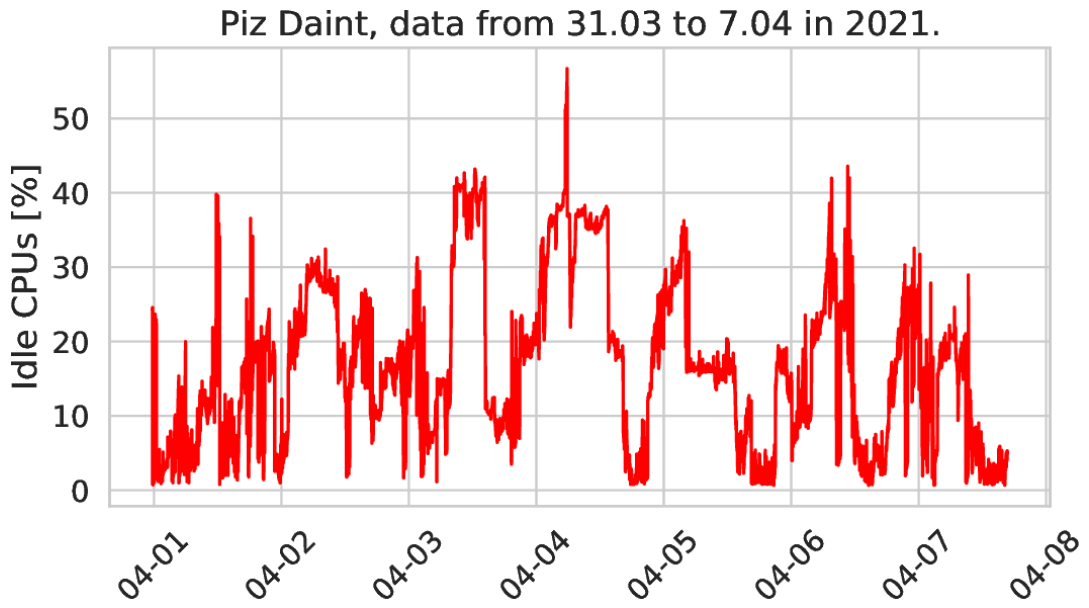
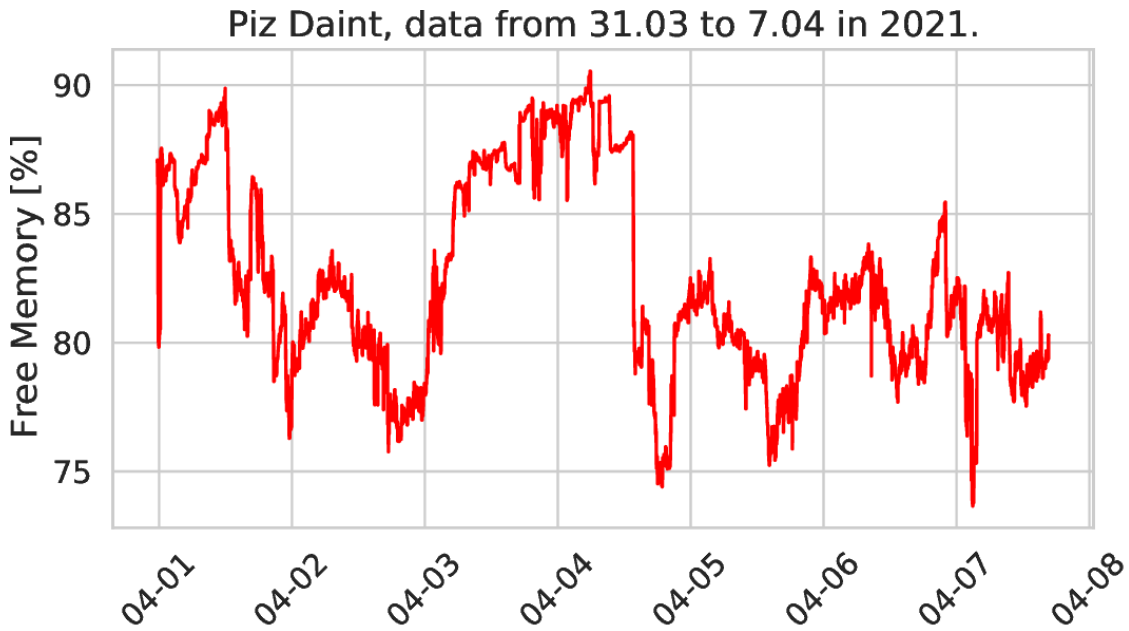
rFaaS: High-Performance Serverless with RDMA

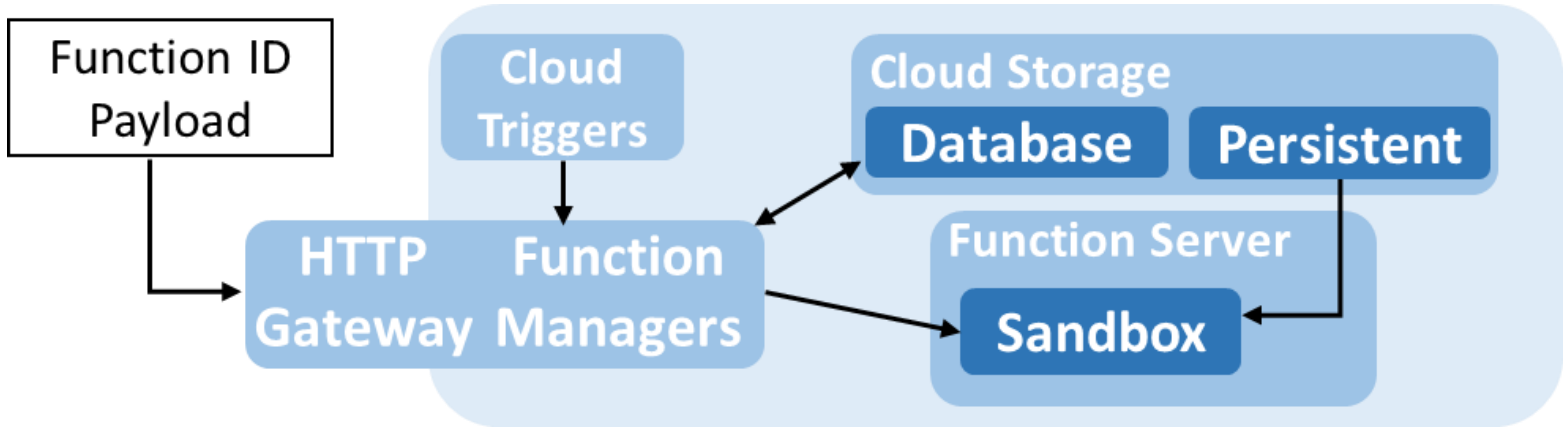
Marcin Copik, Konstantin Taranov, Alexandru Calotoiu, Torsten Hoefler

HPC System Utilization

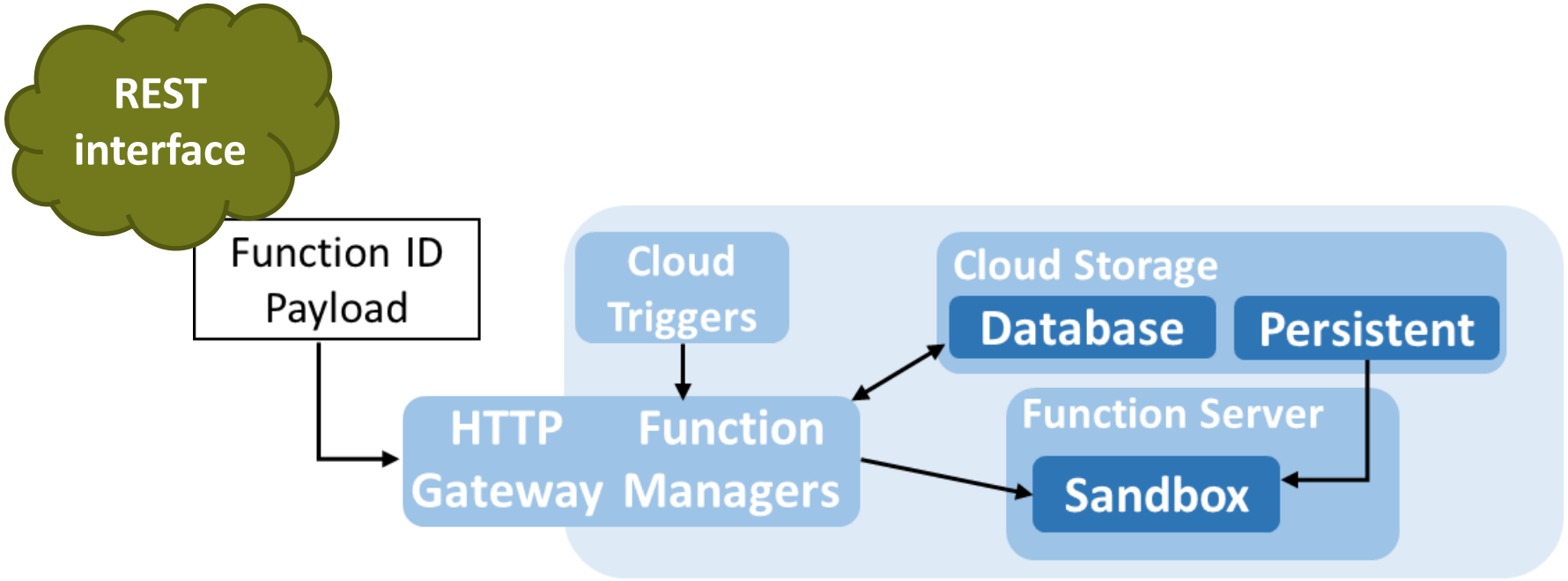




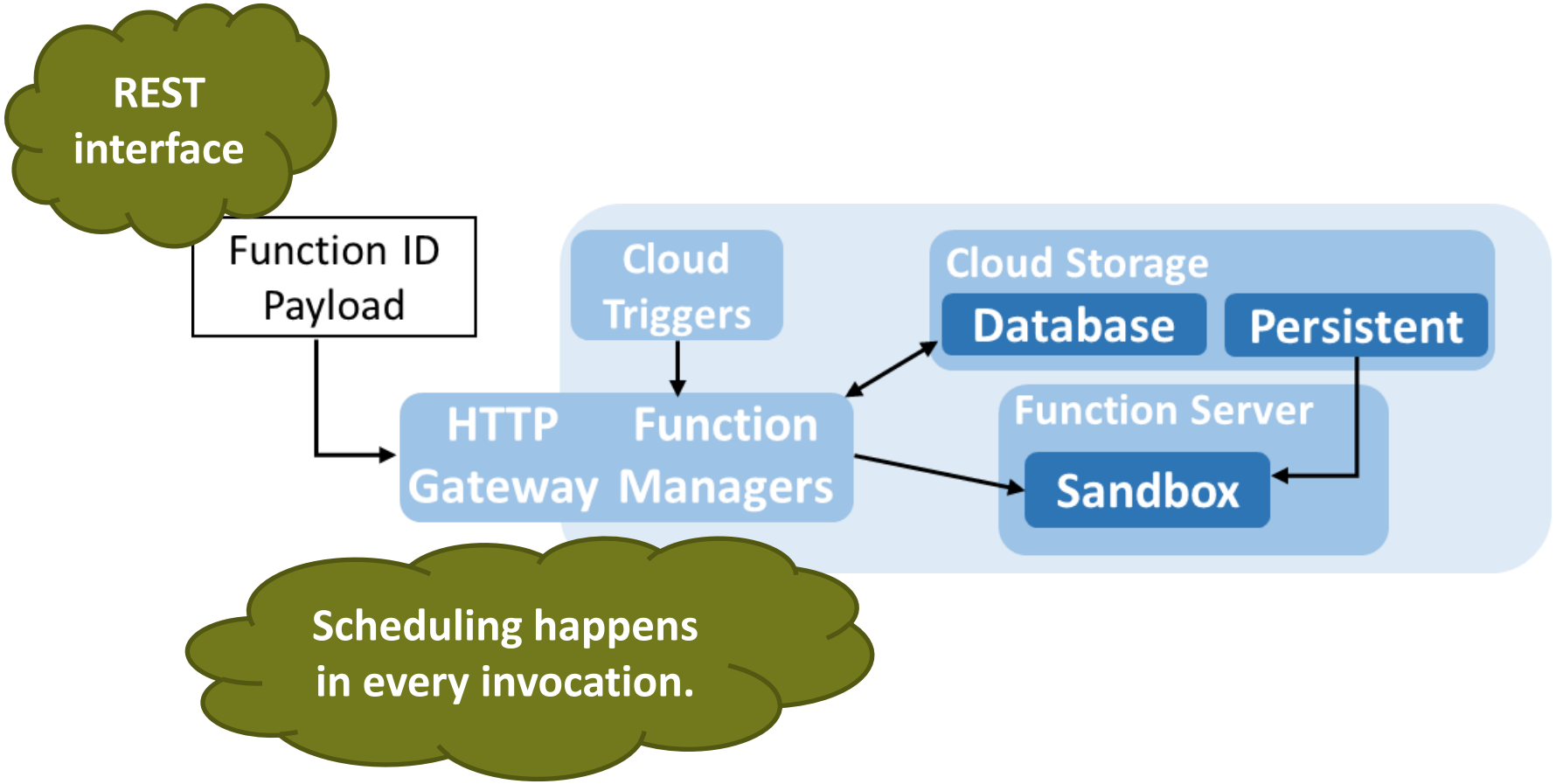
Function-as-a-Service



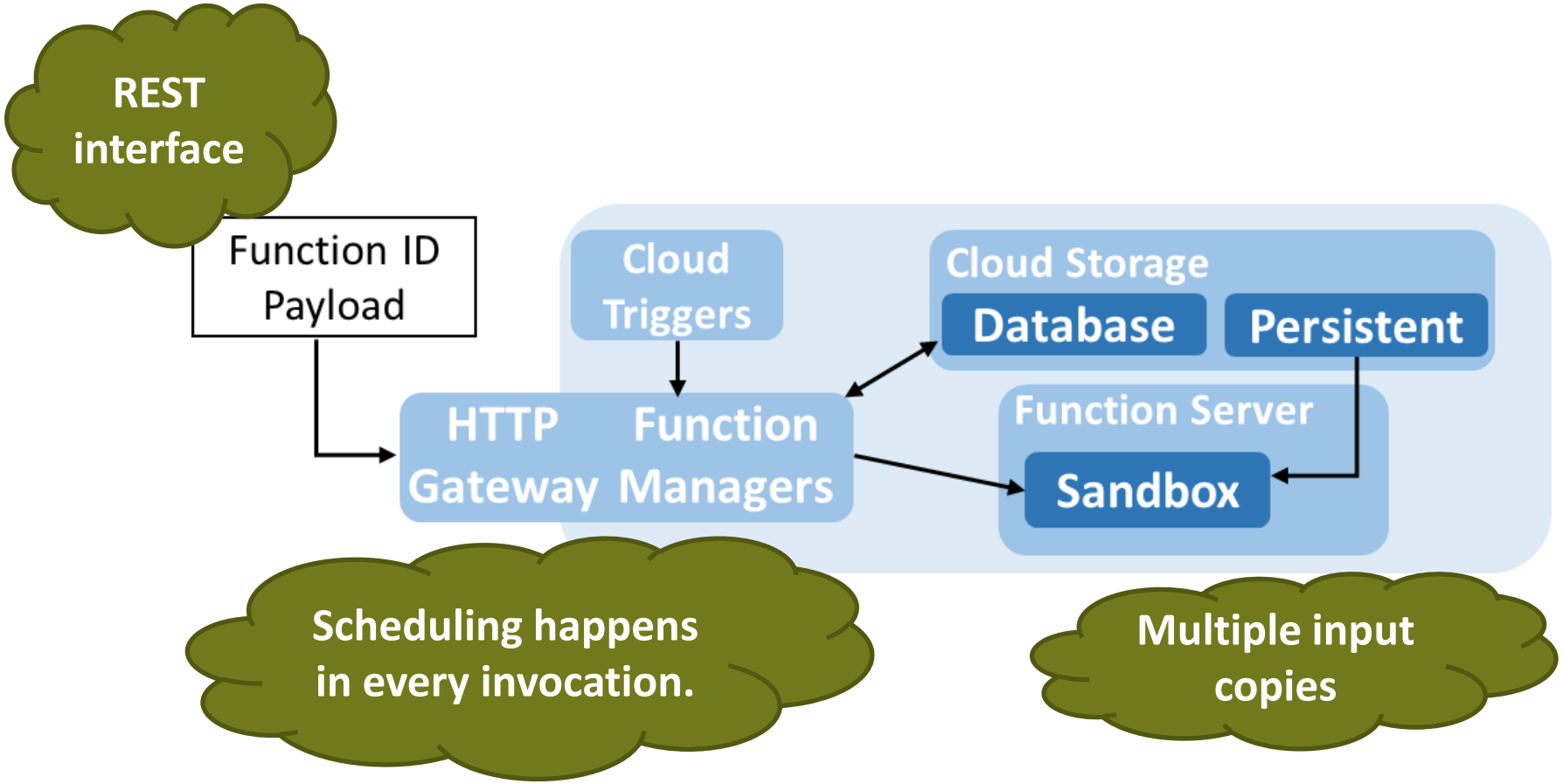
Function-as-a-Service



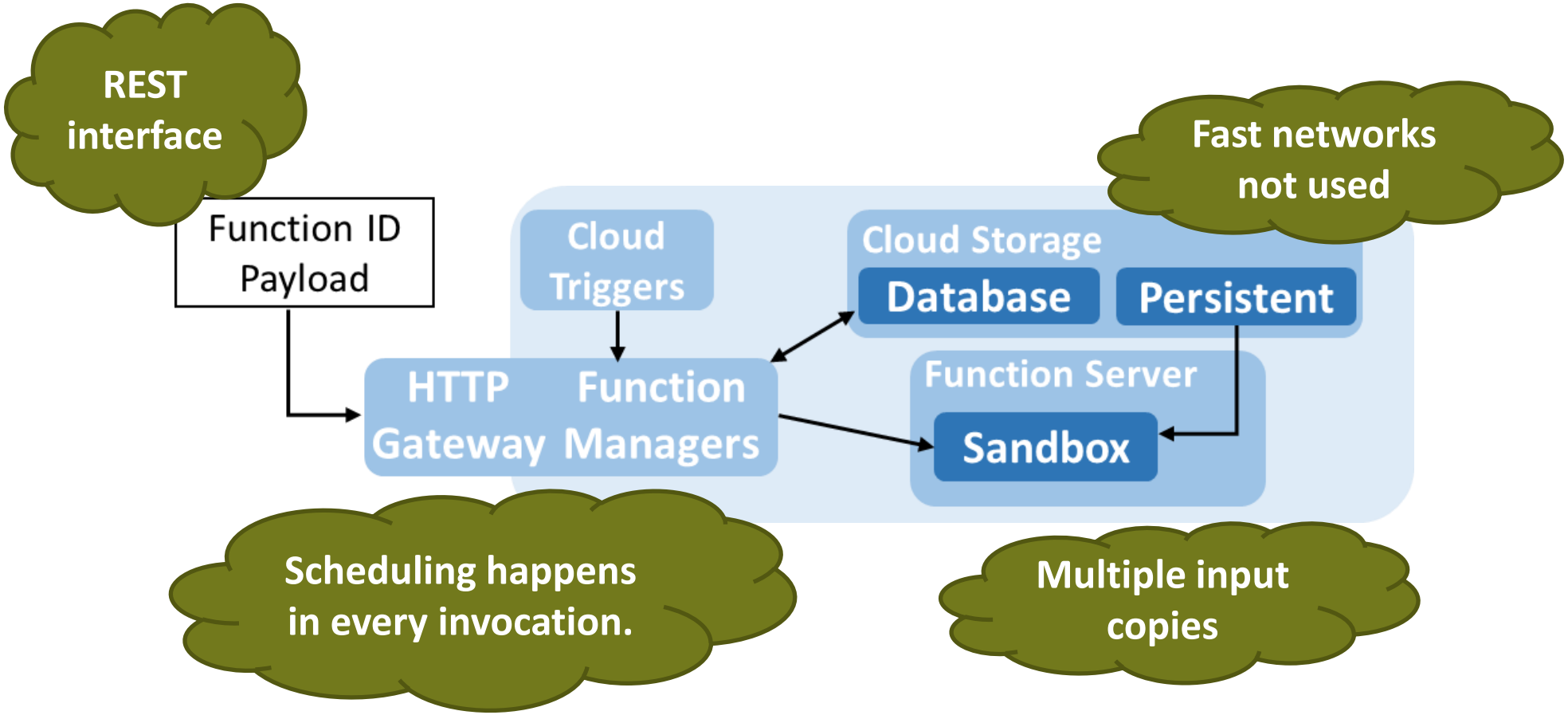
Function-as-a-Service



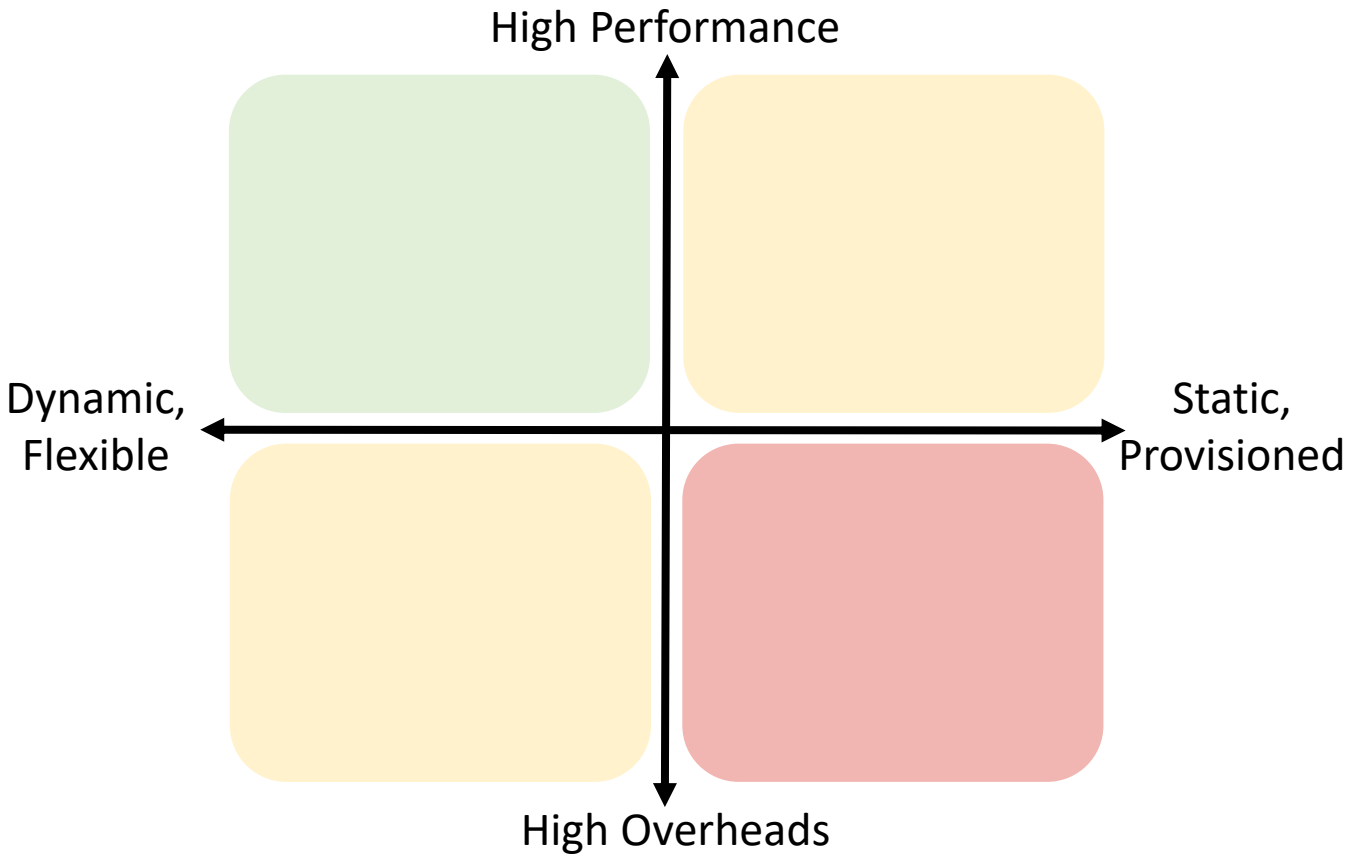
Function-as-a-Service



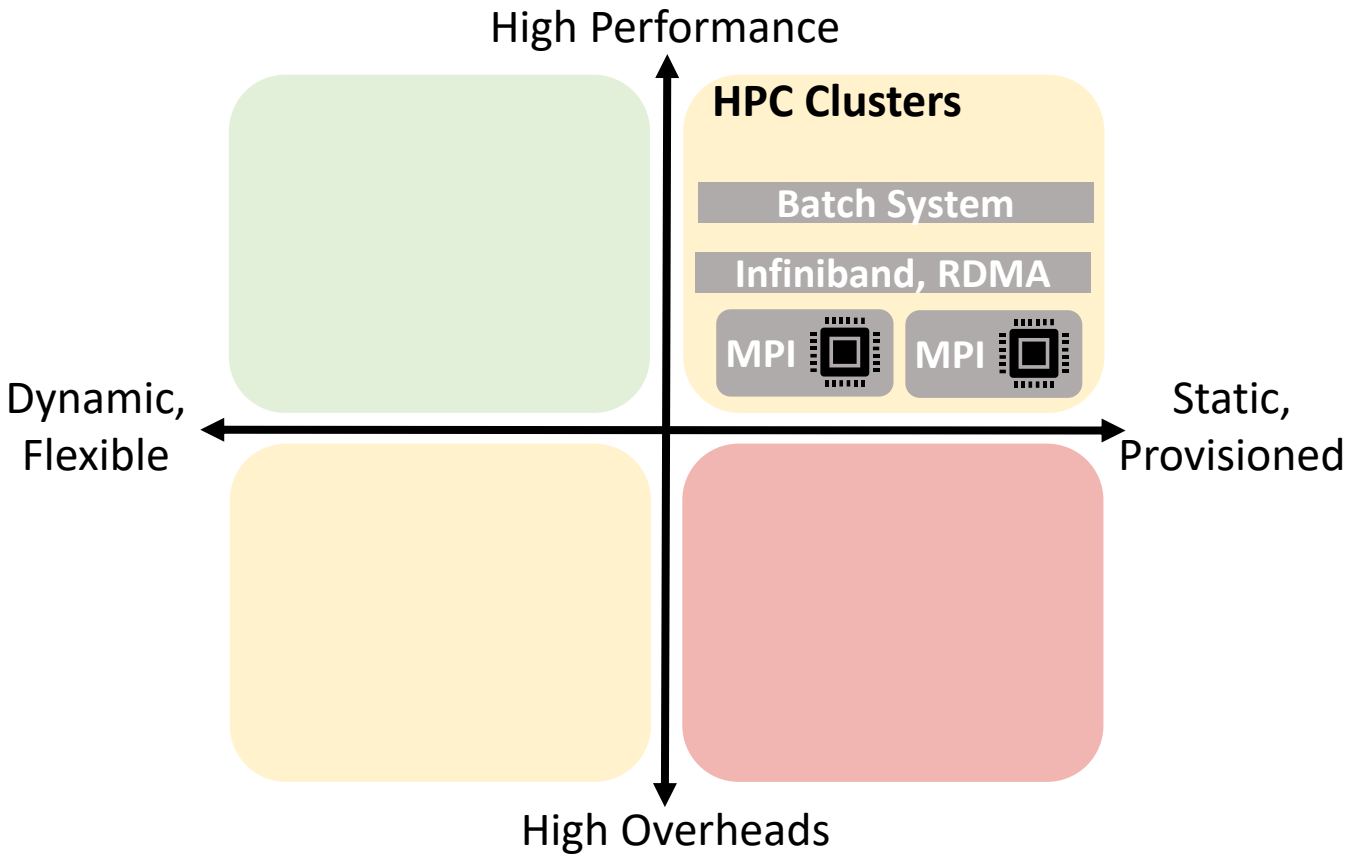
Function-as-a-Service



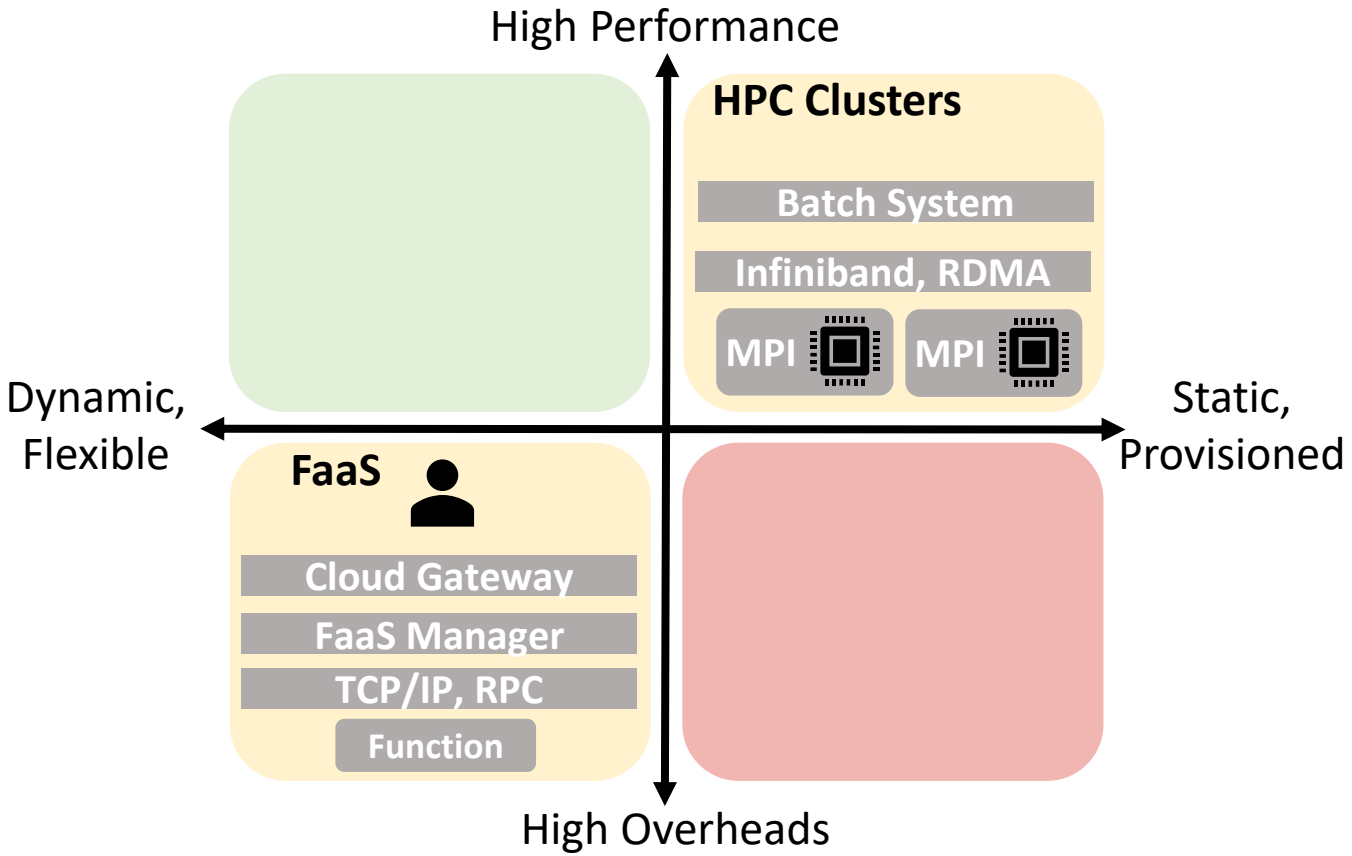
Function-as-a-Service for HPC



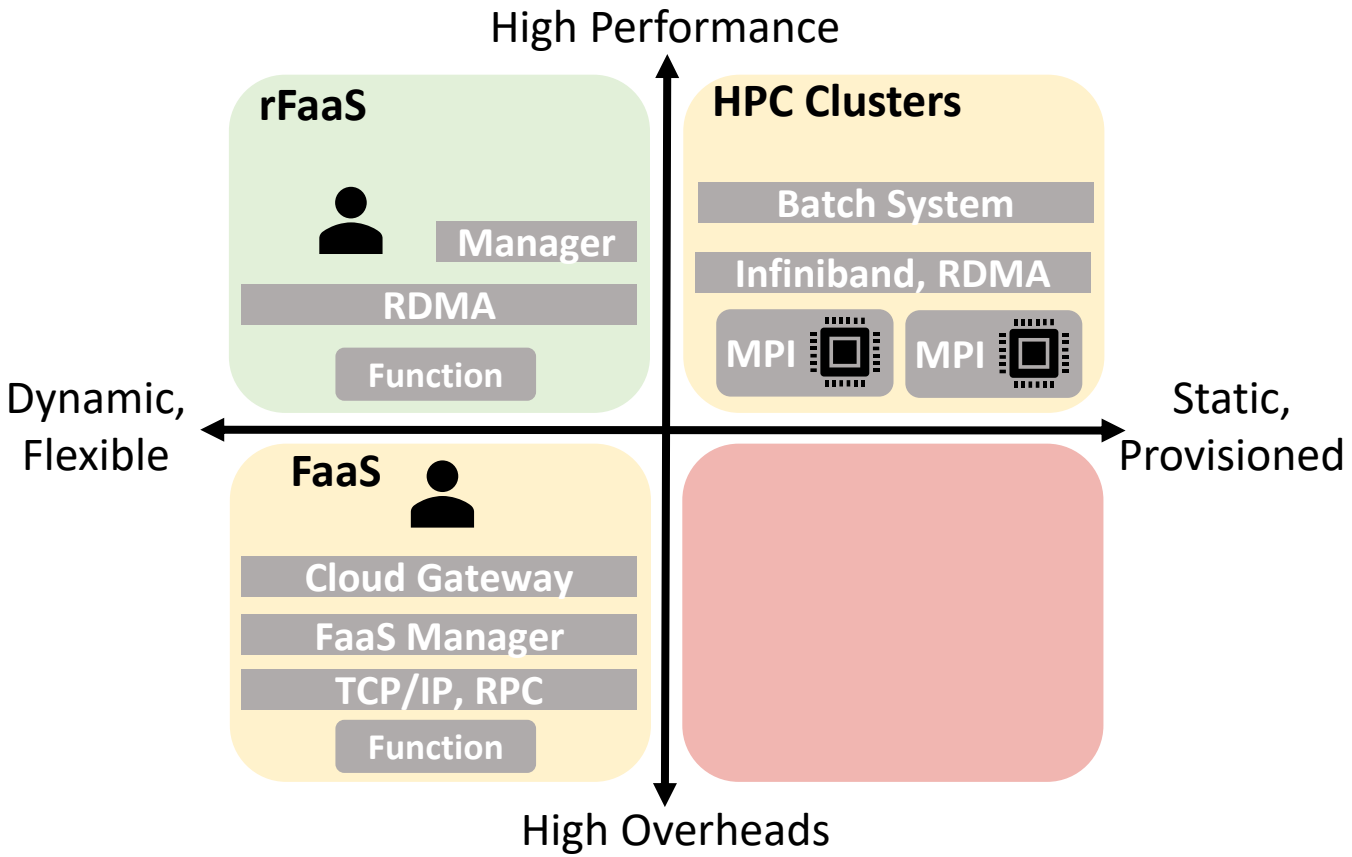
Function-as-a-Service for HPC



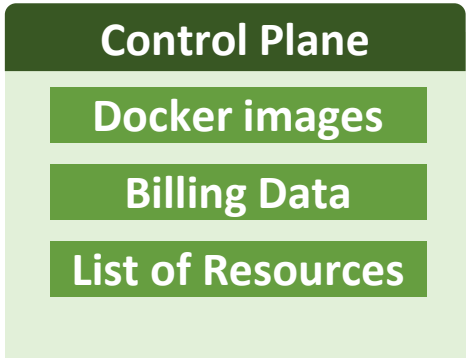
Function-as-a-Service for HPC



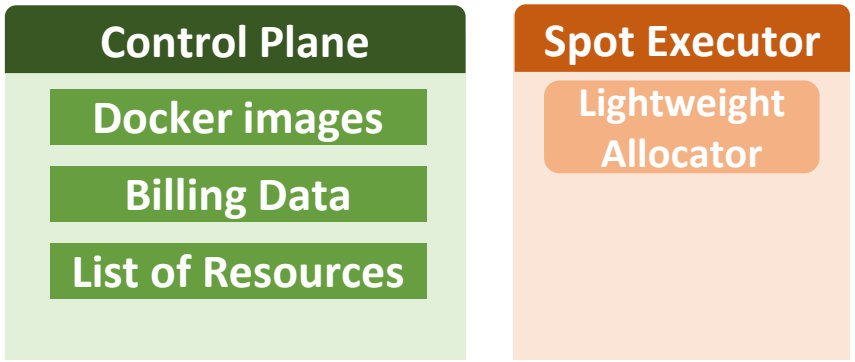
Function-as-a-Service for HPC



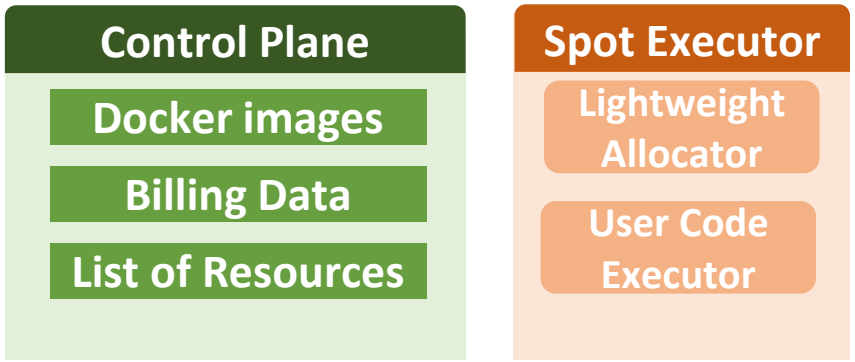
Serverless Platform with RDMA



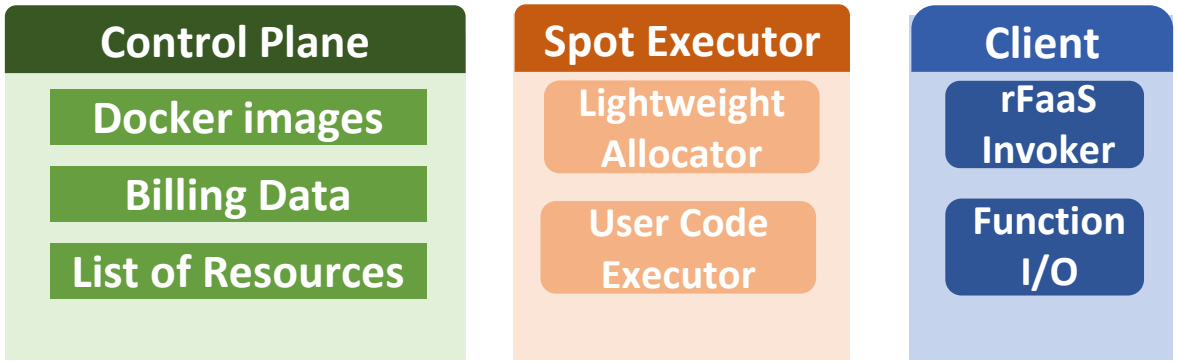
Serverless Platform with RDMA



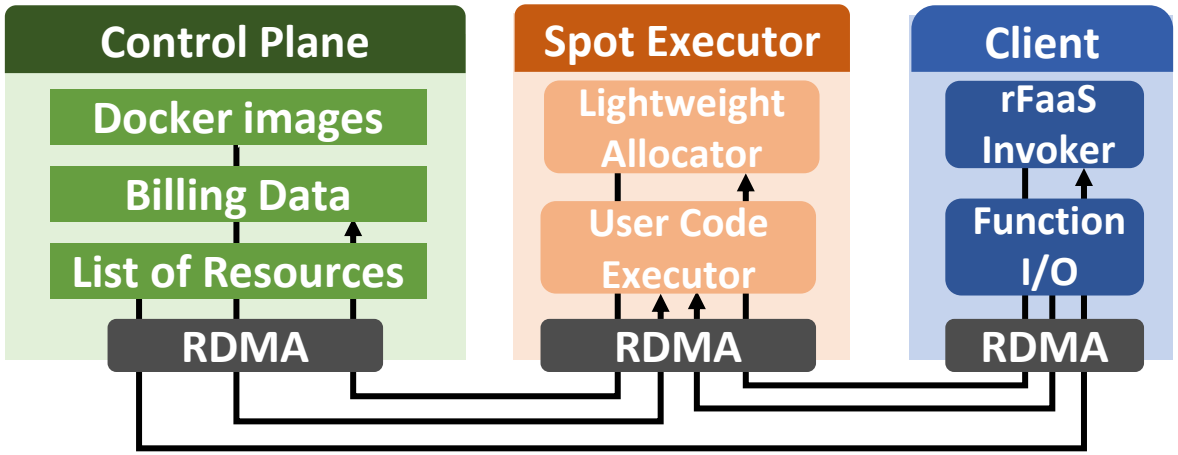
Serverless Platform with RDMA



Serverless Platform with RDMA



Serverless Platform with RDMA



Programming model

```
void compute(int size, options & opts) {
 rfaas::invoker invoker{opts.rnic_device};
 invoker.allocate(opts.lib, opts.size * sizeof(double),
   rfaas::invoker::ALWAYS_WARM_INVOCATIONS);

 auto alloc = invoker.allocator<double>{};
 rfaas::buffer<double> in = alloc.input(2 * size);
 rfaas::buffer<double> out = alloc.output(2 * size);

 auto f = invoker.submit("task", in, size, out);
 local_task(in.data() + size, out.data() + size, size);
 f.get();

 invoker.deallocate();
}
```

Programming model

```
void compute(int size, options & opts) {
 rfaas::invoker invoker{opts.rnic_device};
 invoker.allocate(opts.lib, opts.size * sizeof(double),
   rfaas::invoker::ALWAYS_WARM_INVOCATIONS);

 auto alloc = invoker.allocator<double>{};
 rfaas::buffer<double> in = alloc.input(2 * size);
 rfaas::buffer<double> out = alloc.output(2 * size);

 auto f = invoker.submit("task", in, size, out);
 local_task(in.data() + size, out.data() + size, size);
 f.get();

 invoker.deallocate();
}
```

- Serverless leases

Programming model

```
void compute(int size, options & opts) {
 rfaas::invoker invoker{opts.rnic_device};
 invoker.allocate(opts.lib, opts.size * sizeof(double),
   rfaas::invoker::ALWAYS_WARM_INVOCATIONS);

 auto alloc = invoker.allocator<double>{};
 rfaas::buffer<double> in = alloc.input(2 * size);
 rfaas::buffer<double> out = alloc.output(2 * size);

 auto f = invoker.submit("task", in, size, out);
 local_task(in.data() + size, out.data() + size, size);
 f.get();

 invoker.deallocate();
}
```

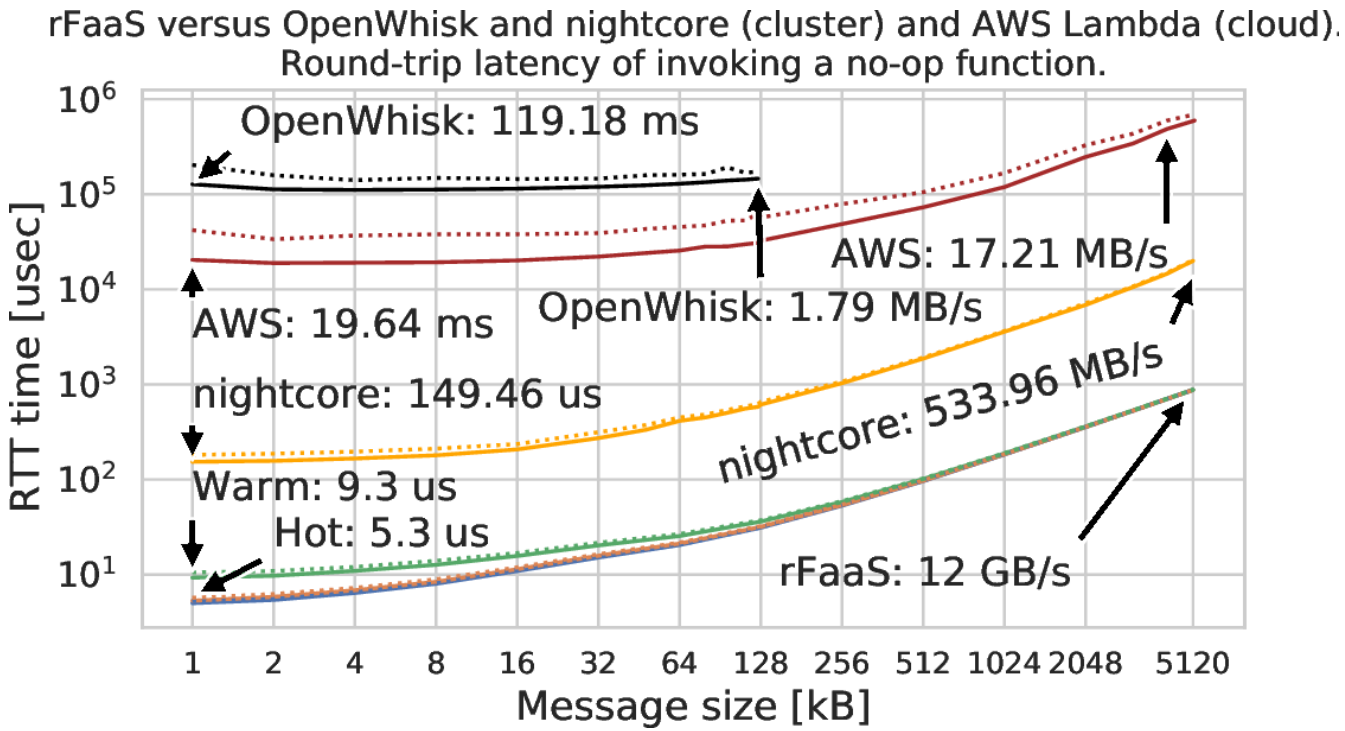
- Serverless leases
- RDMA abstractions

Programming model

```
void compute(int size, options & opts) {
 rfaas::invoker invoker{opts.rnic_device};
 invoker.allocate(opts.lib, opts.size * sizeof(double),
   rfaas::invoker::ALWAYS_WARM_INVOCATIONS);

 auto alloc = invoker.allocator<double>{};
 rfaas::buffer<double> in = alloc.input(2 * size);
 rfaas::buffer<double> out = alloc.output(2 * size);

 auto f = invoker.submit("task", in, size, out);
 local_task(in.data() + size, out.data() + size, size);
 f.get();

 invoker.deallocate();
}
```

- Serverless leases
- RDMA abstractions
- Asynchronous invocations

rFaaS vs FaaS

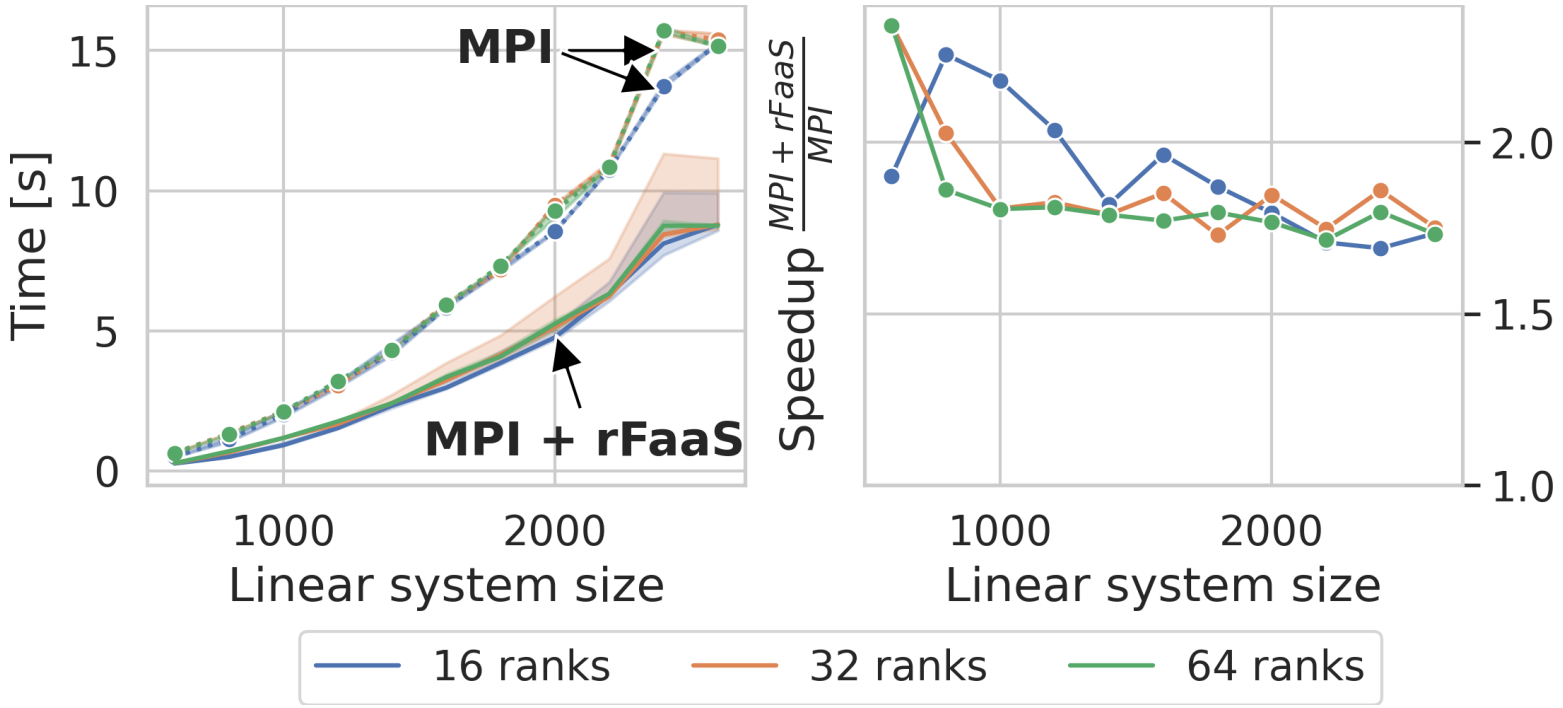


Jacobi Linear Solver

- Bulk Synchronous Parallel (BPS) computation
- Each iteration takes between 1 and 15 milliseconds.
- Optimistic caching of constant matrices.
- Send N elements, receive N/2 elements in each iteration.

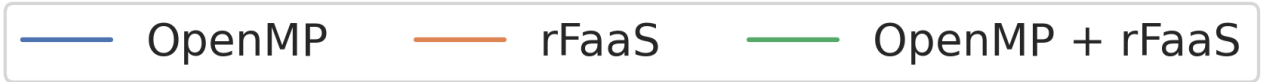
Jacobi Linear Solver

- Bulk Synchronous Parallel (BPS) computation
- Each iteration takes between 1 and 15 milliseconds.
- Optimistic caching of constant matrices.
- Send N elements, receive N/2 elements in each iteration.



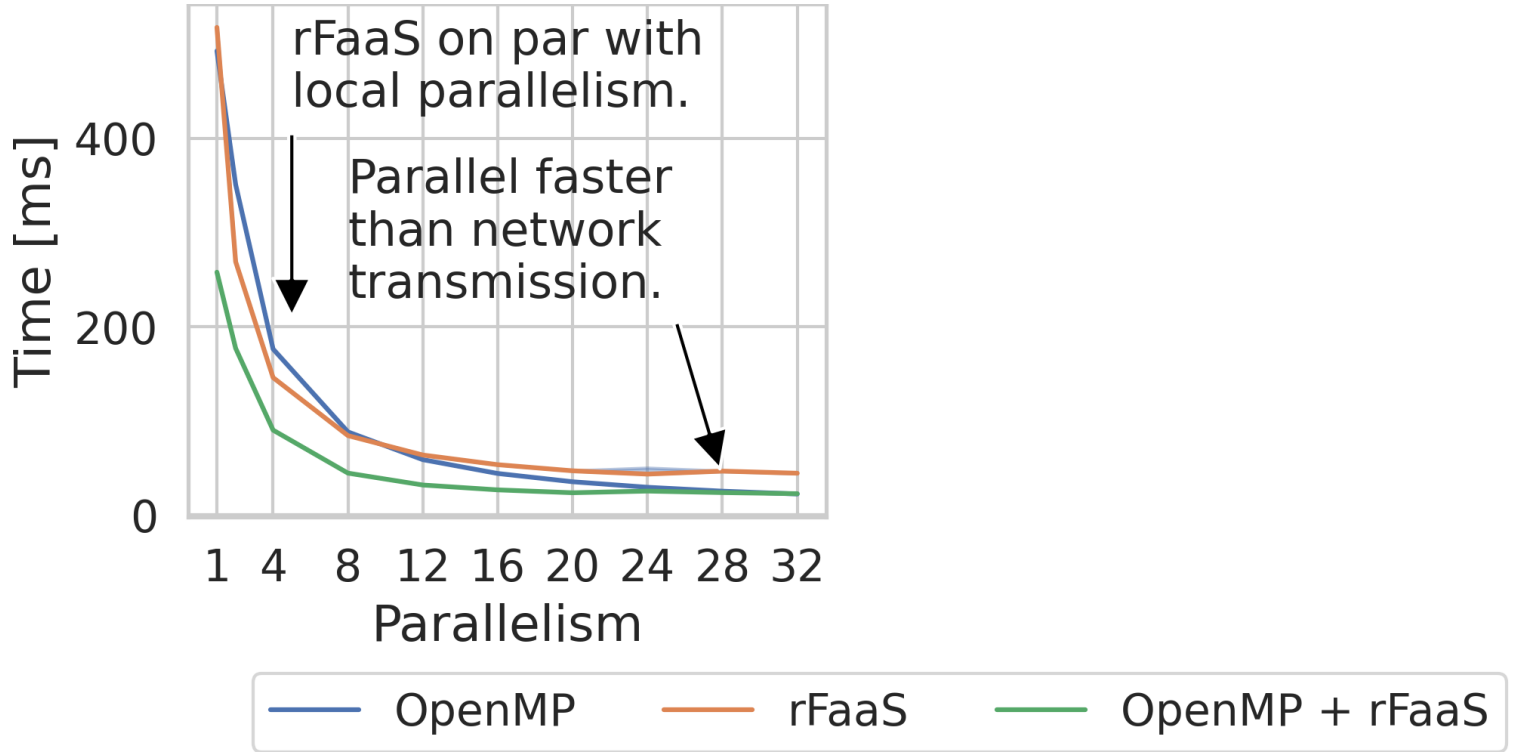
PARSEC: Black-Scholes

- Massively parallel computations
- Offload 50% of work to serverless functions.
- 10M equations, 229M input, 38M output.



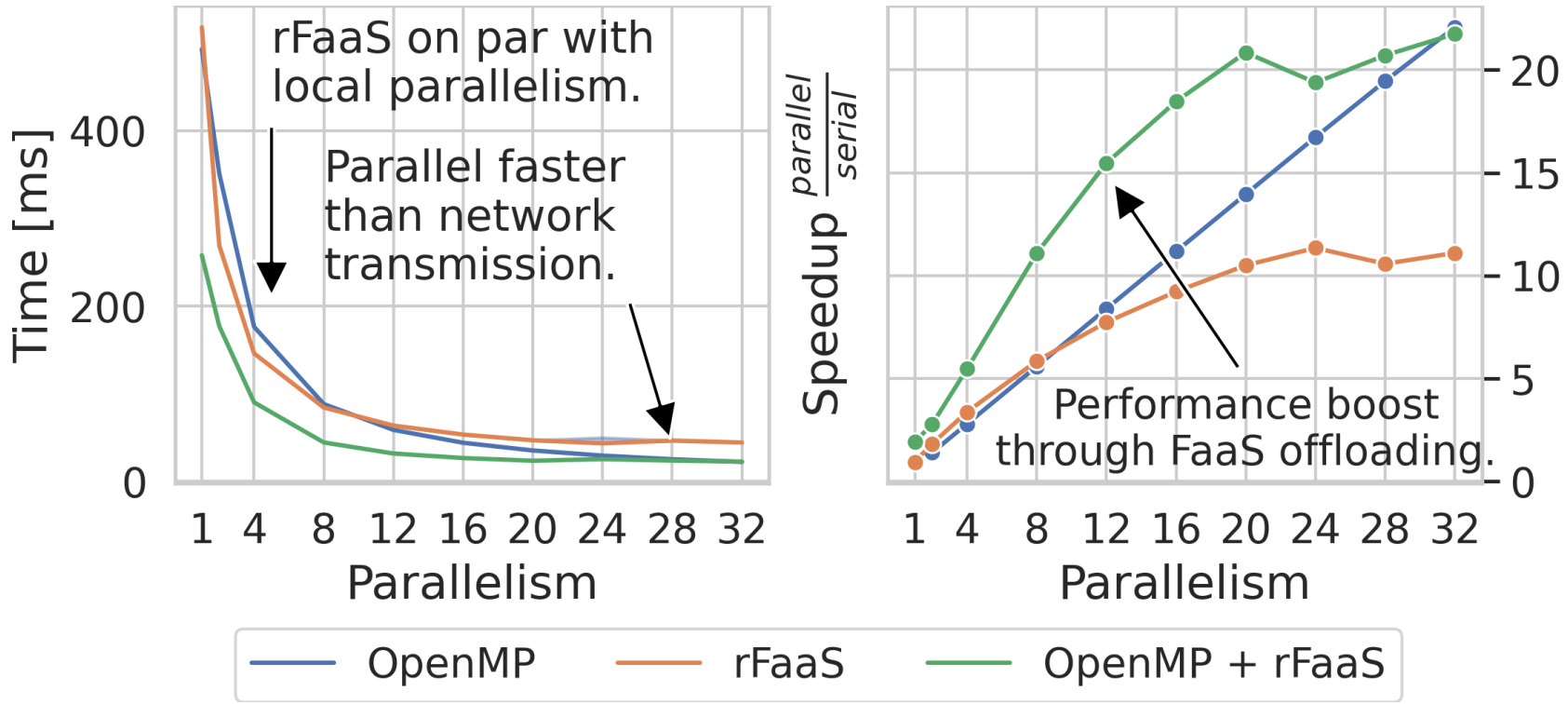
PARSEC: Black-Scholes

- Massively parallel computations
- Offload 50% of work to serverless functions.
- 10M equations, 229M input, 38M output.



PARSEC: Black-Scholes

- Massively parallel computations
- Offload 50% of work to serverless functions.
- 10M equations, 229M input, 38M output.



Ongoing work

- Network support through **libfabrics**.
- HPC containers – Singularity, Sarus.
- Integration of collective operations.
- Compiler and serialization support.

spcl/rFaaS

Paper preprint

https://mcopik.github.io/projects/rfaas/