



Burst Computing: Quick, Sudden, Massively Parallel Processing on Serverless Resources

Daniel Barcelona-Pons, Aitor Arjona, Pedro García-López,
Enrique Molina-Giménez, and Stepan Klymonchuk

USENIX ATC '25
July 7 – 9, 2025
Boston, MA, USA



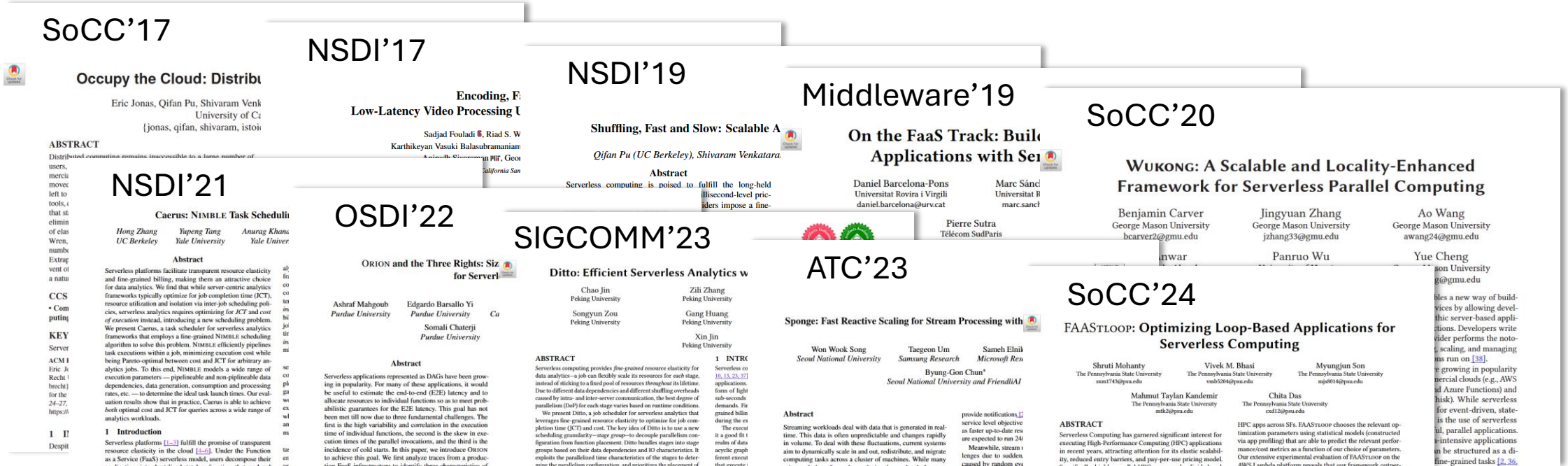
UNIVERSITAT ROVIRA i VIRGILI



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Serverless for Massively Parallel Processing

- FaaS is being used for MPP computations (data/compute-intensive)
  - Data processing and analytics, Machine Learning, …
    - Stream processing, SQL, Monte Carlo simulations, video encoding, code compilation, …
    - Matrix multiplication, data clustering, sorting, batch jobs, …
    - Model tuning, grid search, …



Serverless for Massively Parallel Processing

- Why?

→**Burstability**

- Traditional engines → Slow start-up, rigid, misprovisioning, coarse-grained
- FaaS spawns **thousands** of functions (workers) **in seconds**
- On-demand, fine-grained, scalable, cost-precise, no idle, no-ops
- **Ideal for *bursts*:** short, sudden, massively parallel tasks
- **From cluster-centric to job-centric**

Technology	Total VCPUs	Nodes	Start-up time
EMR Spark	—	6	296 s
		24	431 s
Dataproc	—	6	95 s
		24	113 s
Dask	#¯—	8	184 s
		64	253 s
Ray	#¯—	8	187 s
		64	229 s
Knative (Kubernetes)	—?	960	54 s
OpenWhisk	—?	960	21 s
AWS Lambda (2 GiB)	**—?**	**960**	**6 s**

Serverless for Massively Parallel Processing

- **However!** → FaaS is designed for **event-based applications**
  - Ephemeral, isolated workers
  - Stateless, transient
  - **No guaranteed parallelism**
  - **No worker-to-worker communication**
- Many **bursts are stateful** or need worker collaboration!
  - Sorting, joins, aggregations, iterative/collaborative algorithms
- **FaaS is not Enough**

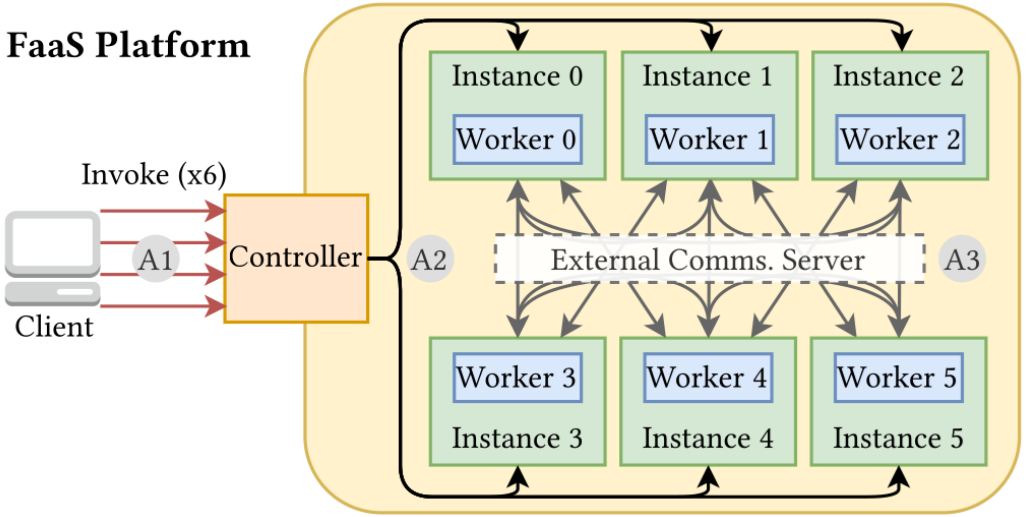
Serverless for Massively Parallel Processing

- Solution? **sidestep limitations** of FaaS?
  - Remote storage for indirect communication (S3, Redis, …)
  - NAT traversal
  - Opportunistic placement of same-function invocations
  - Key-value stores with functions
- Key question:
  - For MPP programs, do we need **per-worker isolation** when they are from same tenant, working on the same job?
  - Why stick to FaaS?
- **Why don’t we evolve FaaS to support bursts?**

FaaS friction points

- Running massively parallel processing (MPP) jobs on FaaS faces three main challenges (frictions):
  - **F1: Worker Isolation**
    Independent invocations (A1, A2) → **skewed start times** and **redundant setup** (code/data duplication).
  - **F2: Job Fragmentation & Orchestration**
    Lack of synchronization → **multiple stage jobs** with intermediate data stored externally.
  - **F3: Huge Data Movement**
    Isolated functions (A3) → **communicate via external channels**, leading to inefficient and **heavy data transfers** (e.g., shuffles).

These frictions hinder the **efficient execution** of both **stateless and stateful** parallel jobs on FaaS platforms.



- ❌ *No coordinated worker spawning*
- ❌ *No Inter-worker communication*
- ❌ *Complex synchronization and orchestration*

Contribution

Burst Computing
A new paradigm for running MPP jobs in the cloud, improving upon FaaS by introducing **group awareness, packing** and **locality exploitation**.

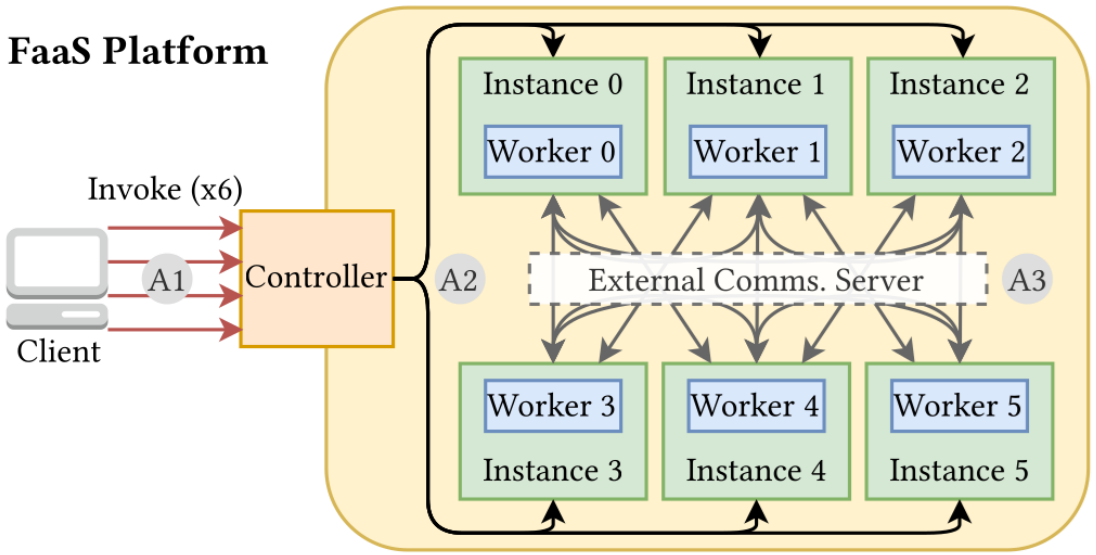
Key Principle of Burst Computing: *Group awareness*

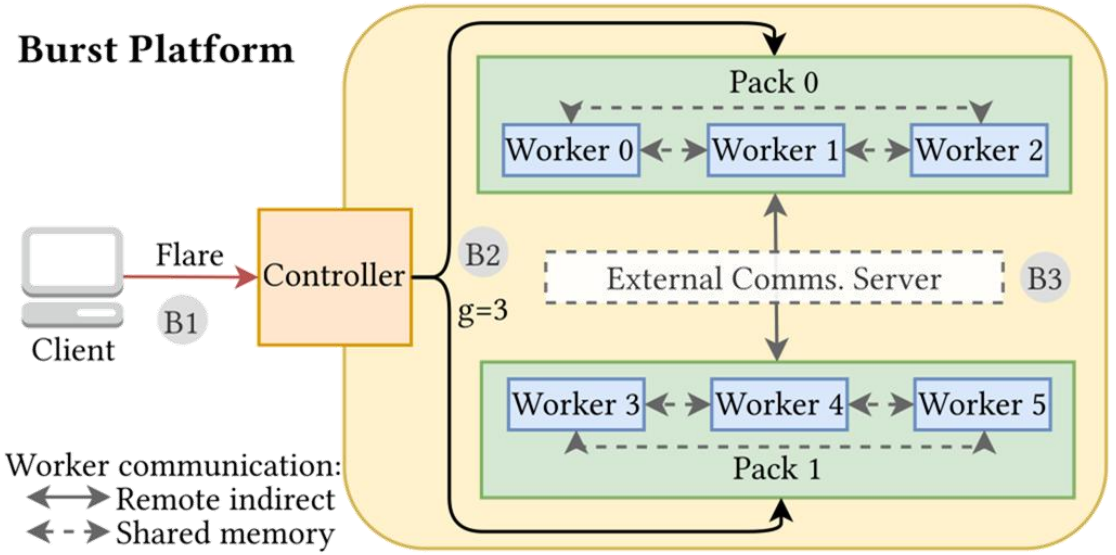
- Raise isolation: function level → **group level**
- An MPP job composed of many workers/functions → Job workers are managed as a **group**

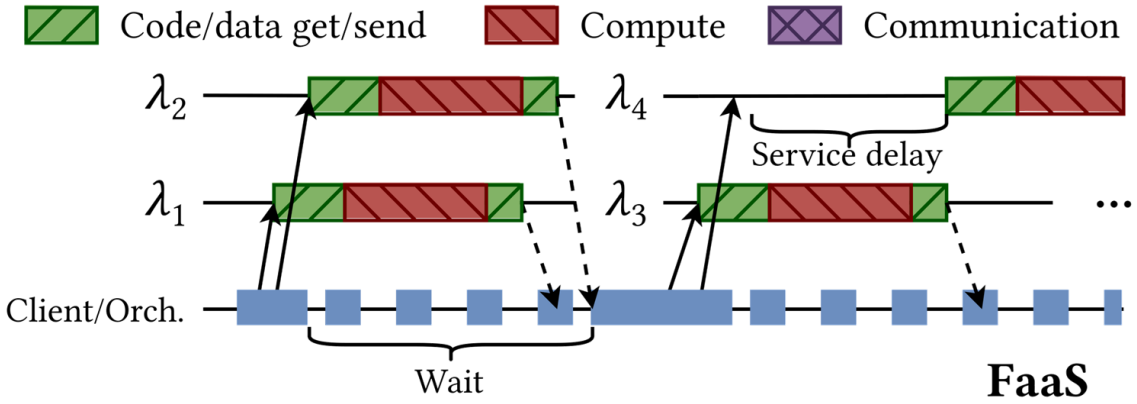
Flare Primitive	Worker Packing	Locality

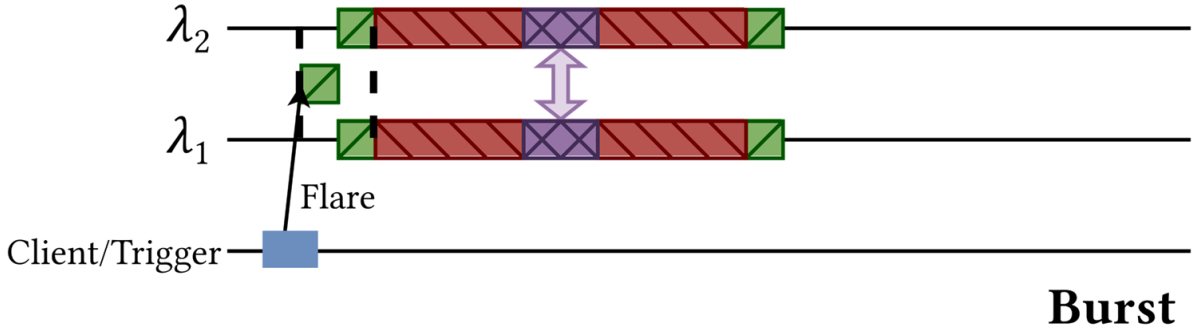
• Group invocation
• Guaranteed parallelism
• Synchronous execution
  • Coordination, collectives
	• Many workers per container
	• Optimize resource allocation
	  • Container creation
	  • Deduplication
		• Worker collaboration
		• Shared memory comms
		• Reduce remote data movement

FaaS vs Burst



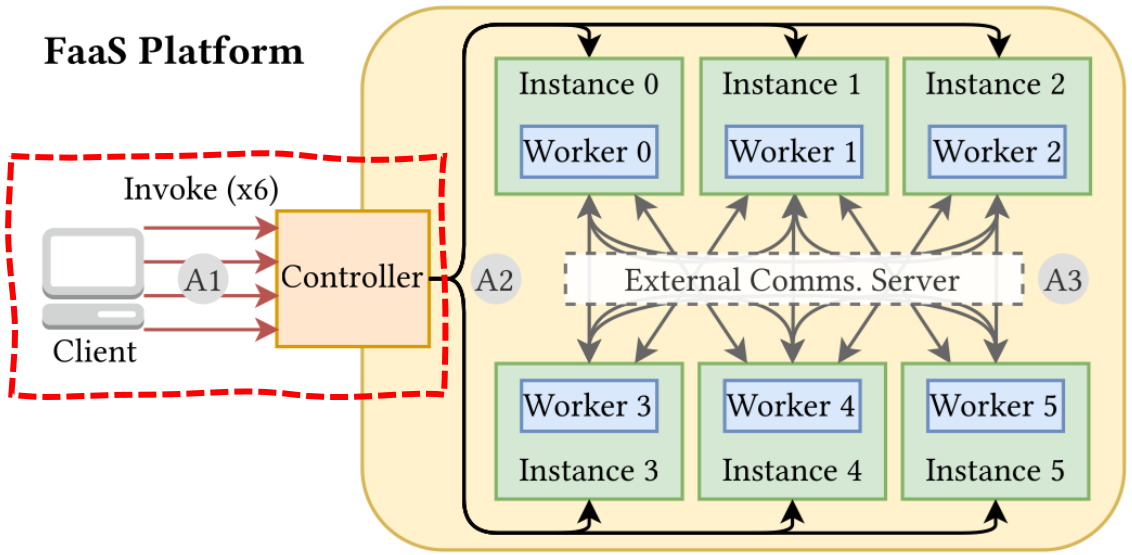


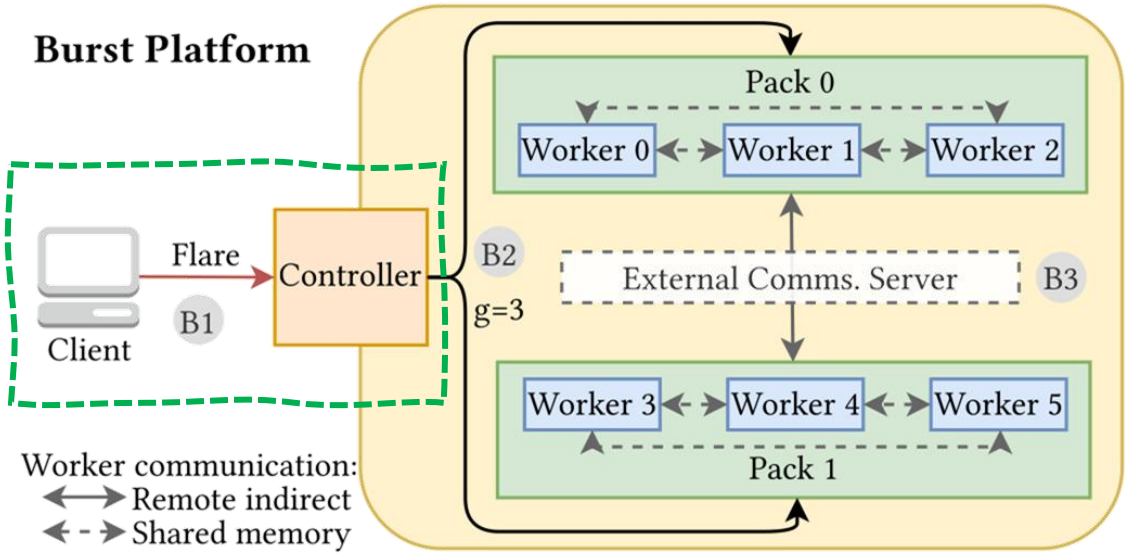


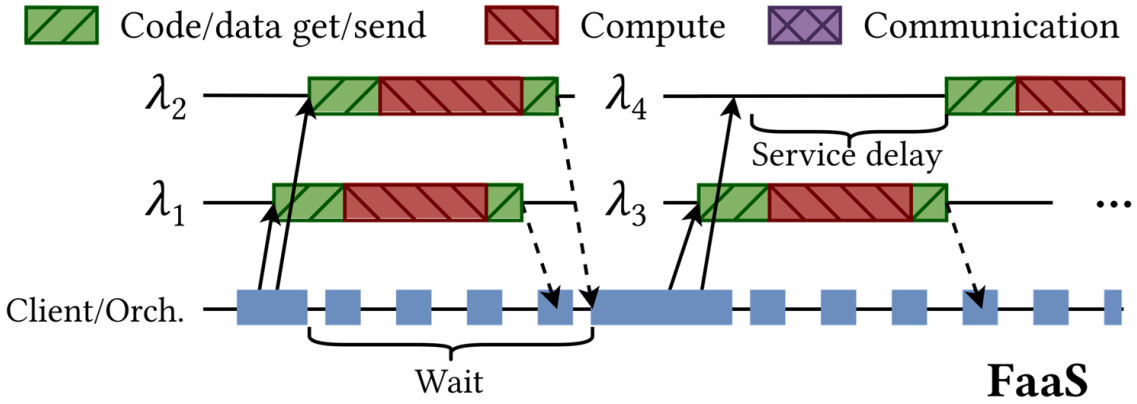


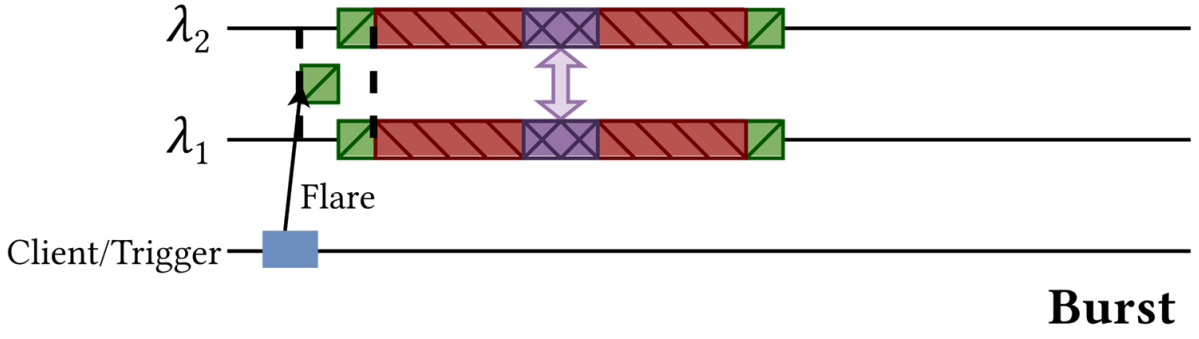
FaaS vs Burst

F1 – Worker Isolation



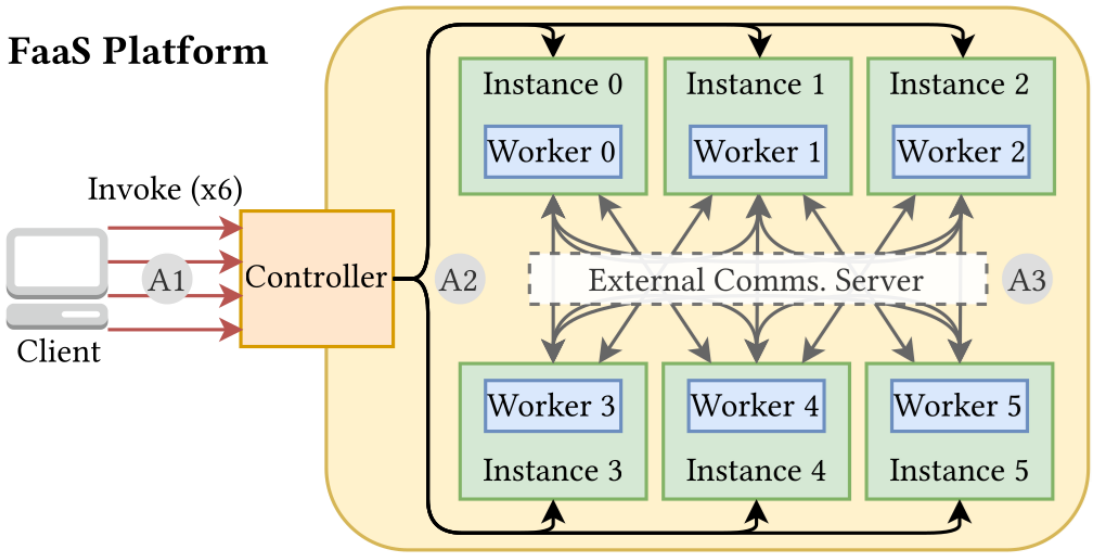


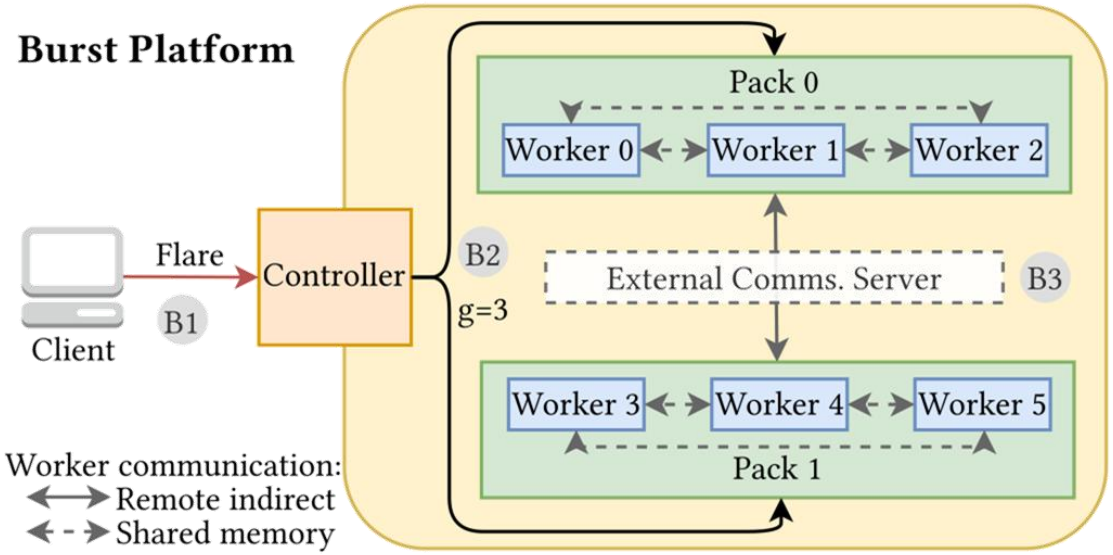


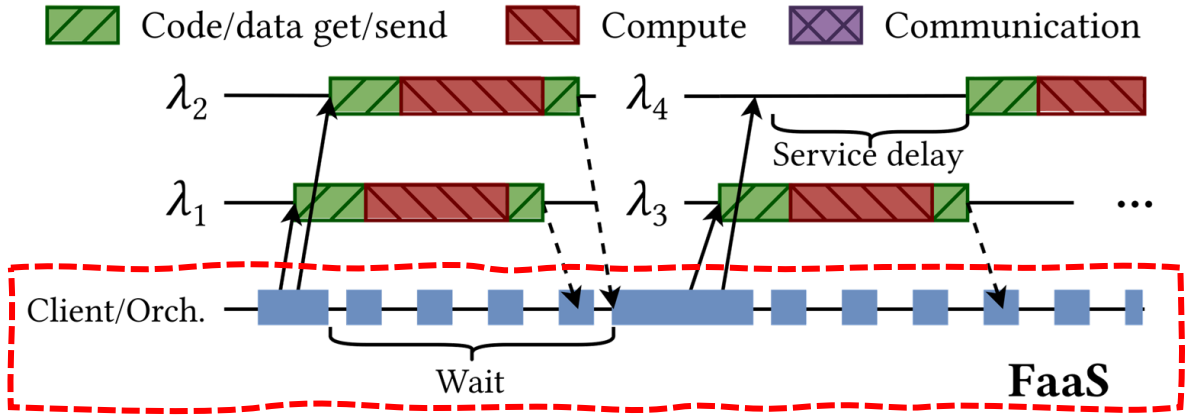


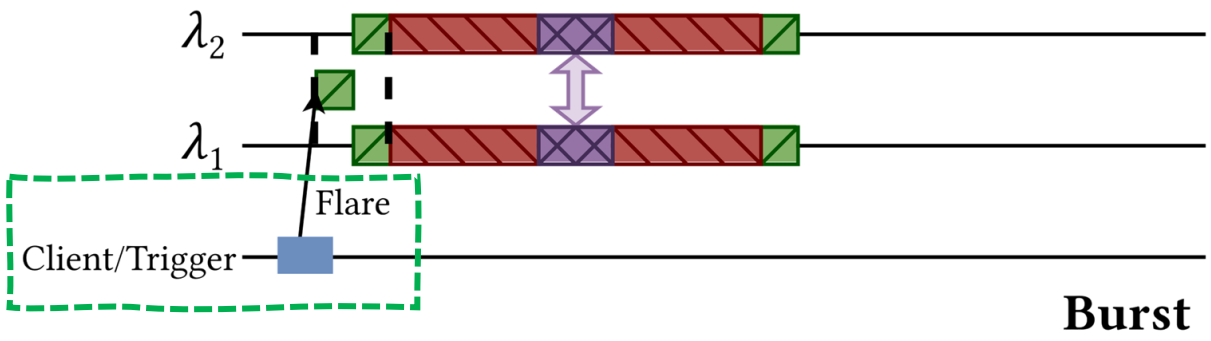
FaaS vs Burst

F2 - Job Fragmentation & Orchestration



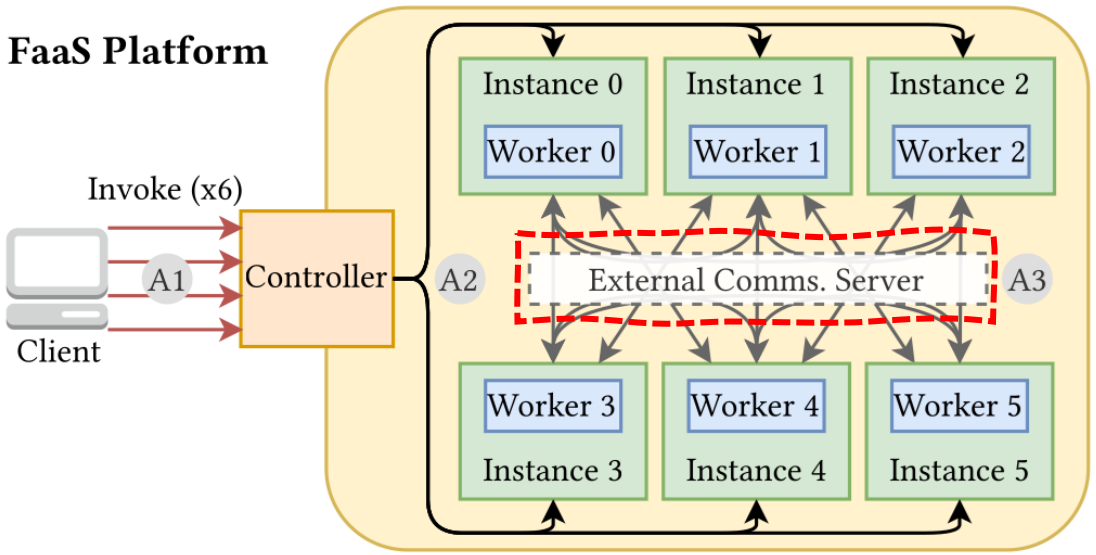


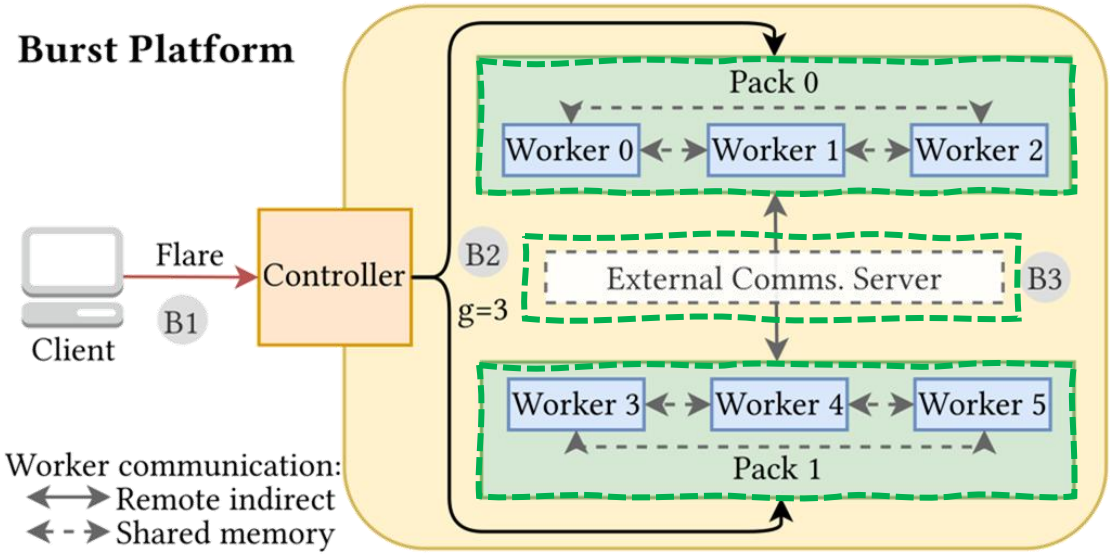


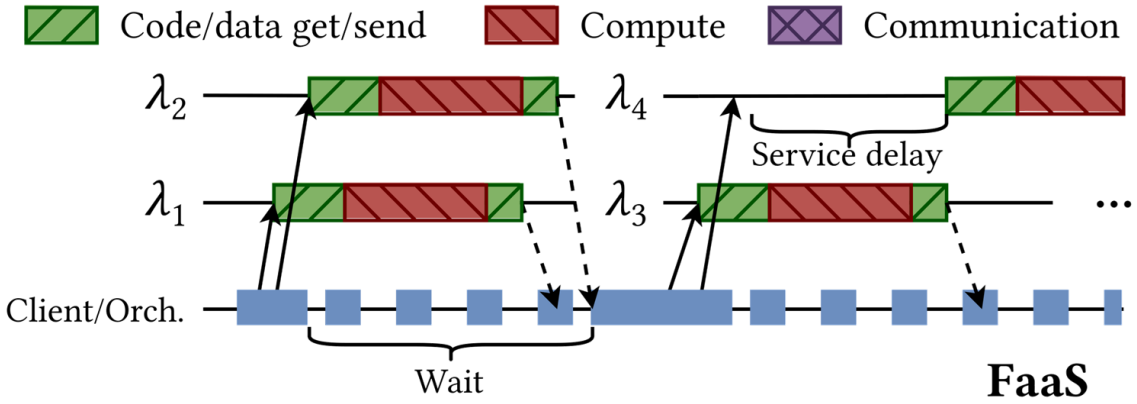


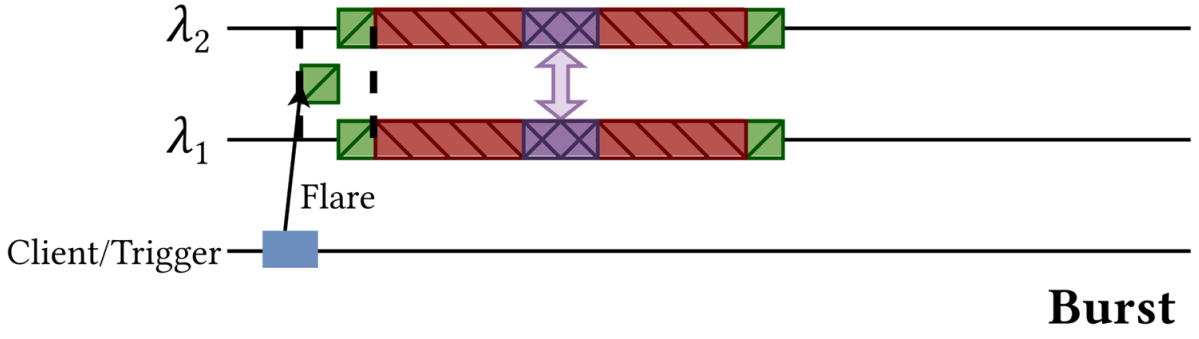
FaaS vs Burst

F3 - Huge Data Movement









Worker packing

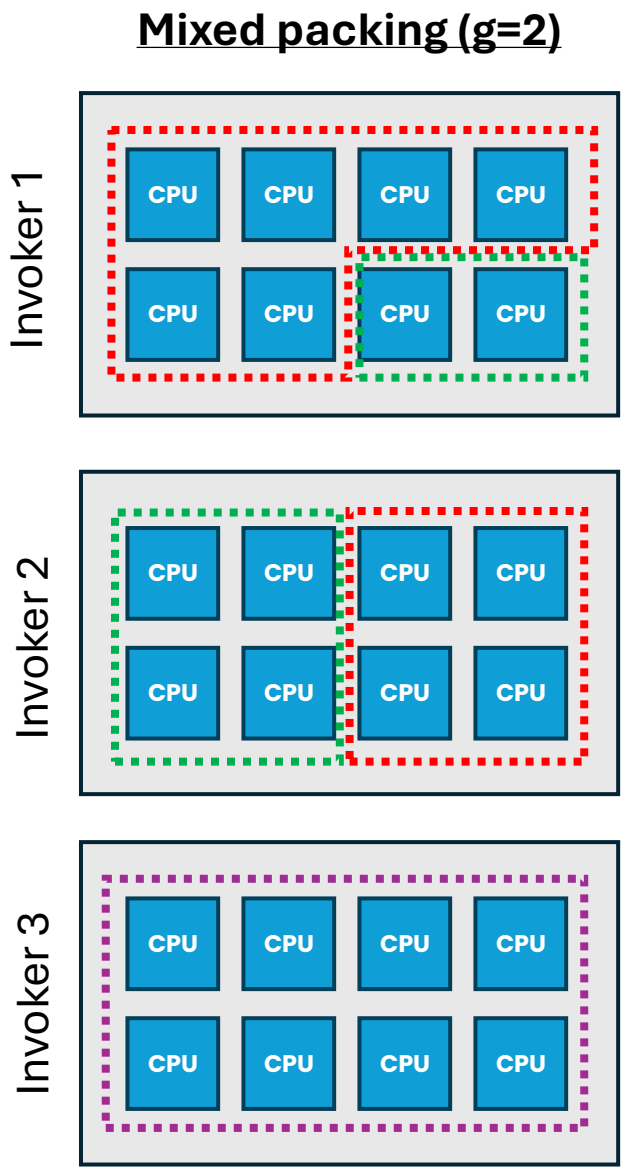
- **Granularity** is the number of workers *per pack*
- What is the correct granularity?
- **Trade-off:** locality vs scheduling complexity

Packing Strategies:

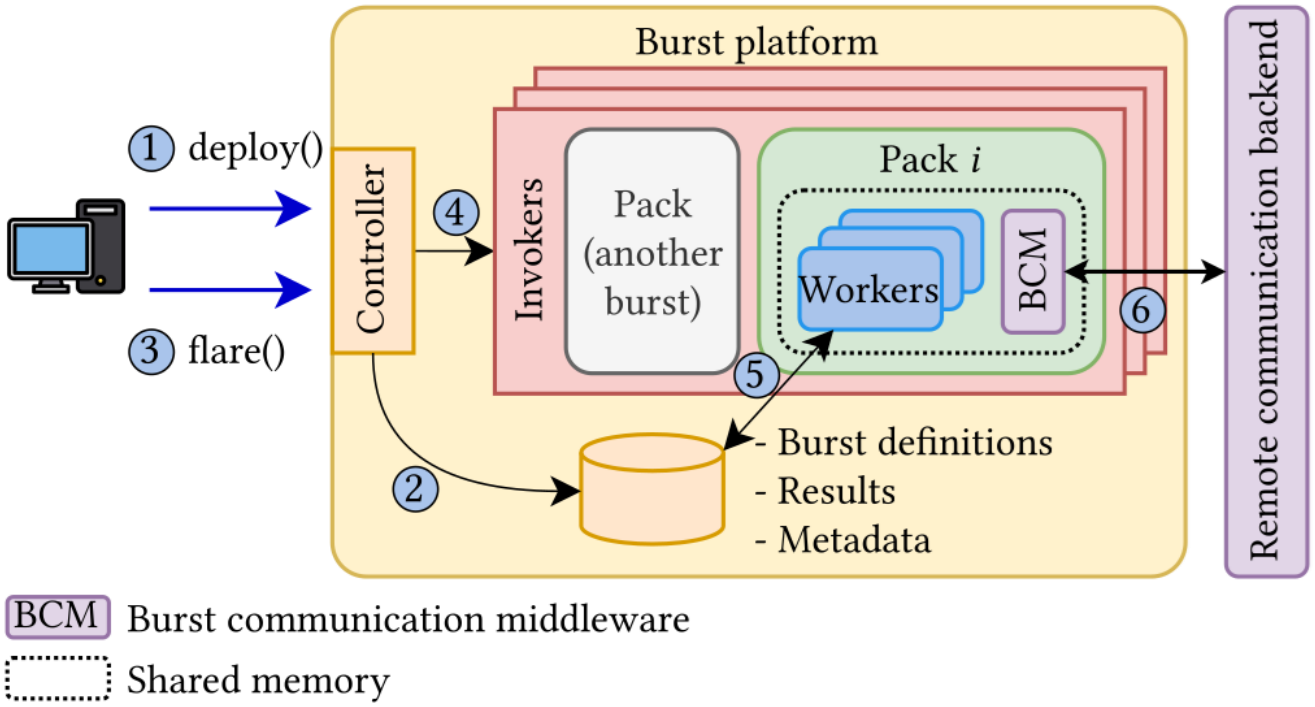
(i) **Heterogeneous:** Packs sized to fit largest possible containers, maximizing locality but risking resource fragmentation.

(ii) **Homogeneous:** Fixed-size packs (e.g., 6 vCPUs), easier scheduling but less worker locality.

(iii) **Mixed:** Fixed-size packs merged on the same machine to balance locality and manageability.



Developing bursts




```rust
fn work(params: Input, burst: &BurstContext) -> Output {
  let num_nodes = params.num_nodes;
  let mut page_ranks = vec![1.0 / num_nodes; num_nodes];
  let mut sum = vec![0.0; num_nodes];
  let adjacency_matrix = get_adjacency_matrix(&params);
  while err < ERROR_THRESHOLD {
    page_ranks = burst.broadcast(page_ranks, ROOT_WORKER);
    for (node, links) in graph {
      for link in links {
        sum[*link] += page_ranks[*node] / out_links(*node);
      }
    }

    let reduced_ranks = burst.reduce(sum, |vec1, vec2| {
      vec1.zip(vec2).map(|(a, b)| a + b).collect()
    });

    if burst.worker_id == ROOT_WORKER {
      err = calculate_error(&page_ranks, &reduced_ranks);
      page_ranks = reduced_ranks;
    }

    err = burst.broadcast(err, ROOT_WORKER);
    reset_sums(&mut sum);
  }
  Output { page_ranks }
}
```

Evaluation

- **Goal** → Compare burst computing to current FaaS for the three friction points
  - **F1 –** Worker Isolation
  - **F2 –** Job Fragmentation & Orchestration
  - **F3 –** Huge Data Movement

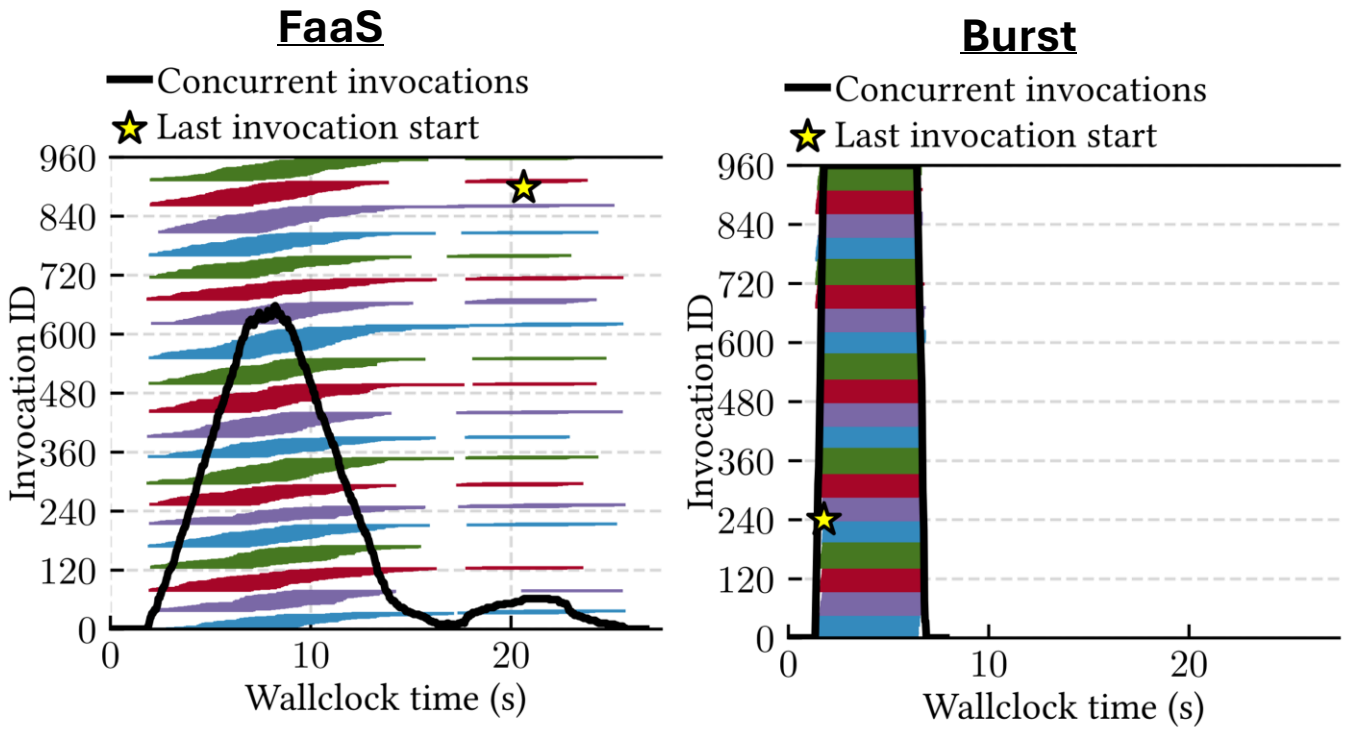
- **Setup** → Experiments run on Amazon Web Services, on EKS clusters
- Compare Apache OpenWhisk with Burst-enabled OpenWhisk

Full evaluation details in the paper!

Worker parallelism

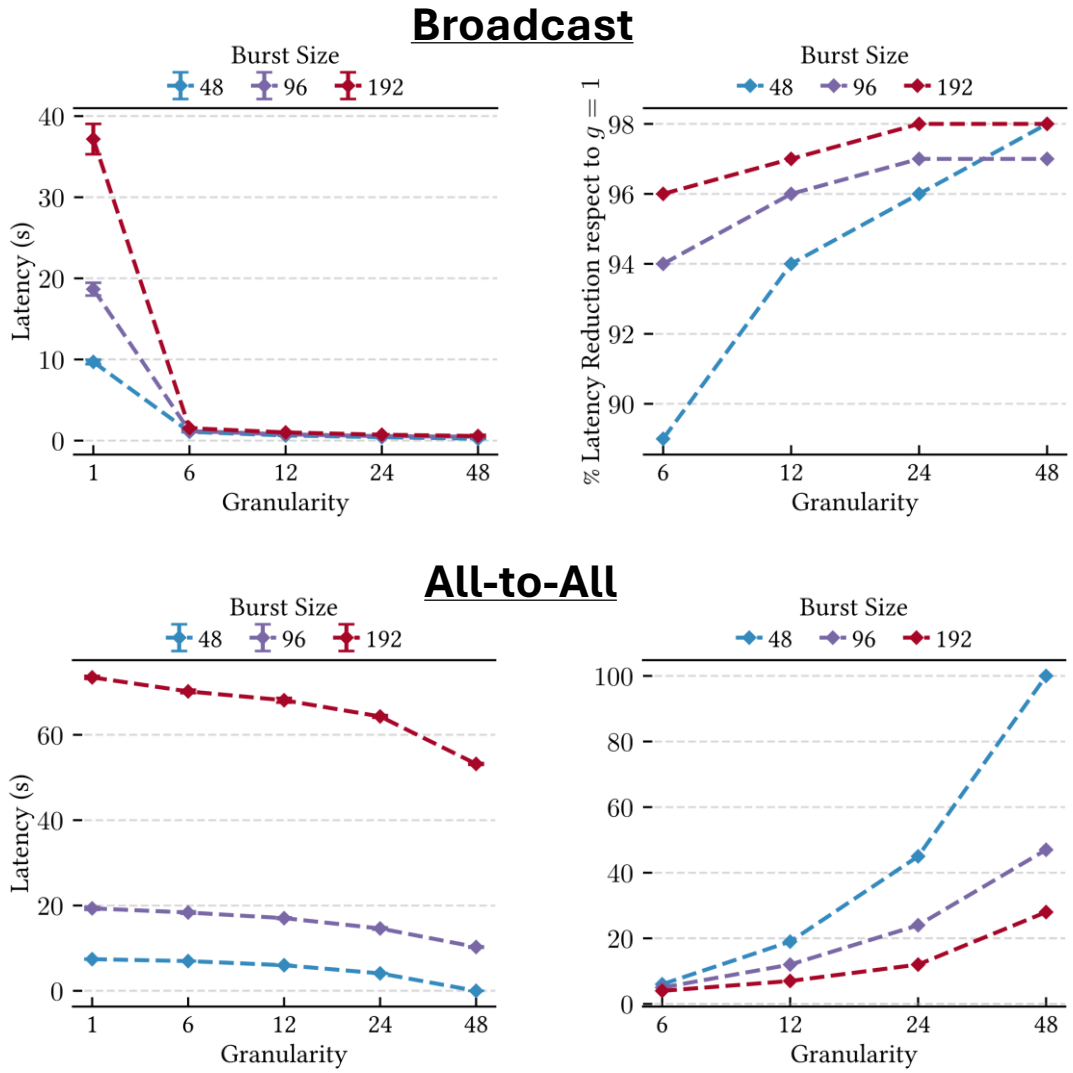
- Burst computing improves worker simultaneity
- Reduced **the start-up time by 43× compared to FaaS**

Technology	Total VCPUs	Nodes	Start-up time
EMR Spark	—	6	296 s
		24	431 s
Dataproc	—	6	95 s
		24	113 s
Dask	#¯—	8	184 s
		64	253 s
Ray	#¯—	8	187 s
		64	229 s
Knative (Kubernetes)	—?	960	54 s
OpenWhisk	—?	960	21 s
AWS Lambda (2 GiB)	—?	960	6 s
Burst Computing	**—?**	**960**	**1.7 s**



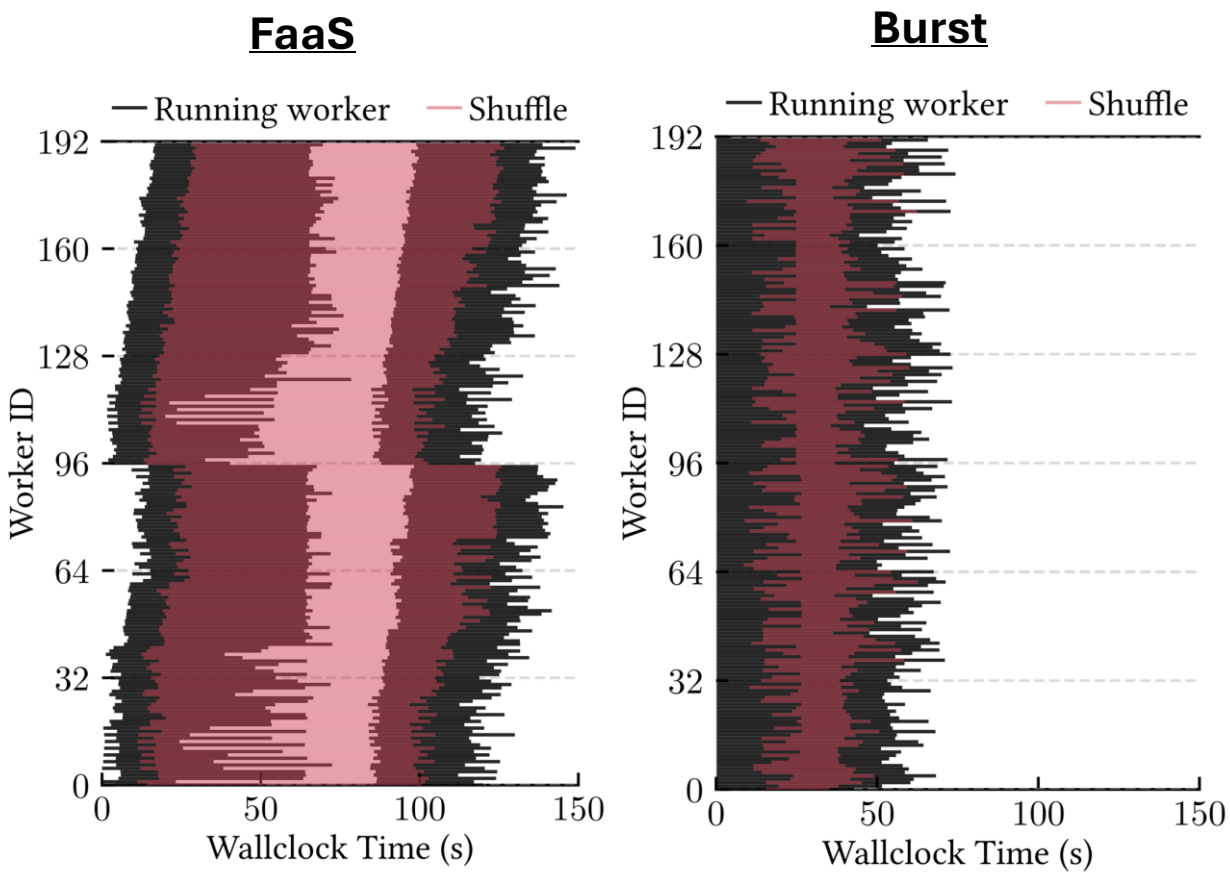
Data movement savings in collectives

- **Reduced remote data movement based on granularity**
- Higher granularity leads to fewer, larger packs, **reducing remote communication**
  - **Broadcast** up to 98% latency reduction
  - **All-to-all** latency reductions at g=48 are
    - 100% for burst size 48 (1 pack)
    - 50% for 96 (2 packs)
    - 25% for 192 (4 packs)



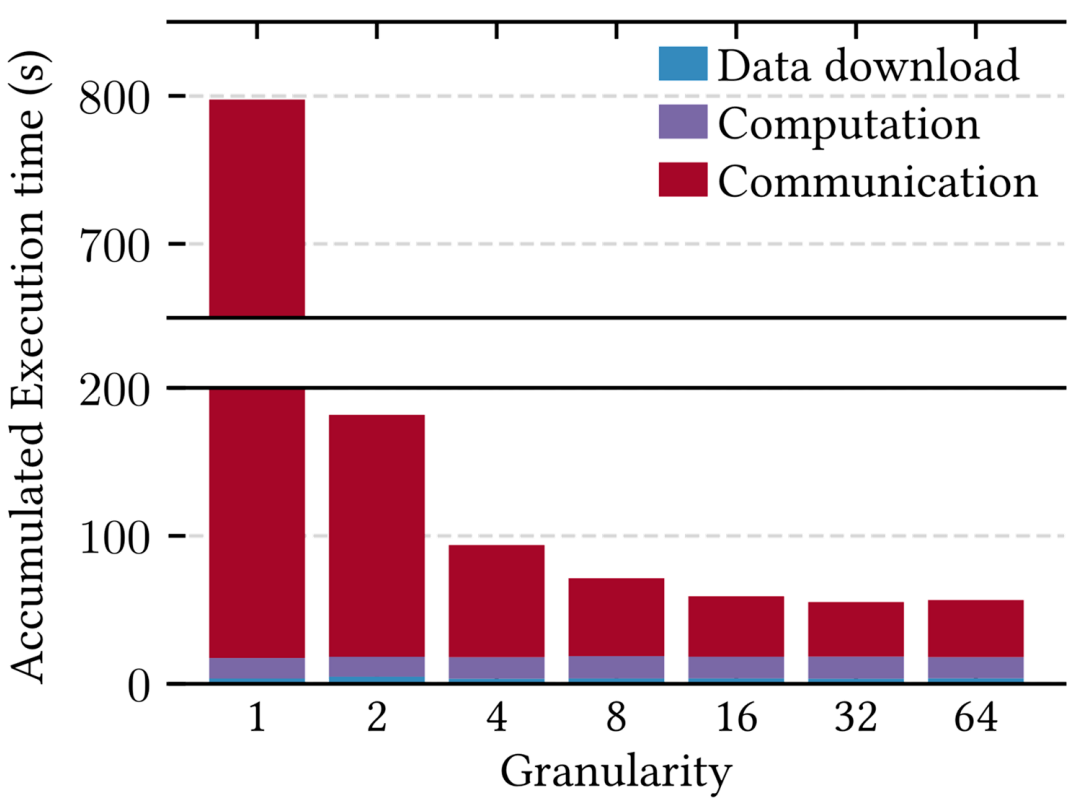
TeraSort application

- **Stateful problem**
  - **FaaS** → Two rounds of function invocations (map and reduce) and shuffles data via object storage.
  - **Burst computing** → Single flare, using locality-aware all-to-all collective to handle the shuffle phase.
- Burst computing achieves a **2× speed-up**



PageRank application

- **Reduce communication overhead**
  - **FaaS MapReduce** → Poor performance with many short, sequential stages required for iterations.
  - Communication dominates total time
  - More locality reduces remote communication
  - **Remote data** → *g=1* to *g=64*, a **98.5% reduction**.
  - **Execution time** → improves 13× at *g=64* compared to *g=1*.



Conclusion

FaaS is not enough!

- FaaS for **Burstability**?
- MPP / Bursts are stateful!
- Three frictions:
  - **F1 - Worker isolation**
  - **F2 - Job fragmentation**
  - **F3 - Data movement**

Burst computing

- **Group awareness**
- **Worker packing → locality**
- Solve frictions:
  - **F1 - Flare**: group invocation
  - **F2 - Parallelism** allows sync worker collaboration
  - **F3 - Locality**-aware communication reduces data movement





Thank you!

daniel.barcelona@urv.cat

https://github.com/Burst-Computing

Funded by the European Union

NEARDATA
neardata.eu

EXTRACT
A distributed data-mining software platform for extreme data across the compute continuum
extract-project.eu

This work is partially funded by the Horizon Europe programme under grant agreements No. 101092644 (Neardata) and No. 101093110 (Extract); and by the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU (PRTR and MRR framework) through the CLOUDLESS UNICO I+D CLOUD 2022.

UNIVERSITAT ROVIRA i VIRGILI

BSC Barcelona Supercomputing Center
Centro Nacional de Supercomputación