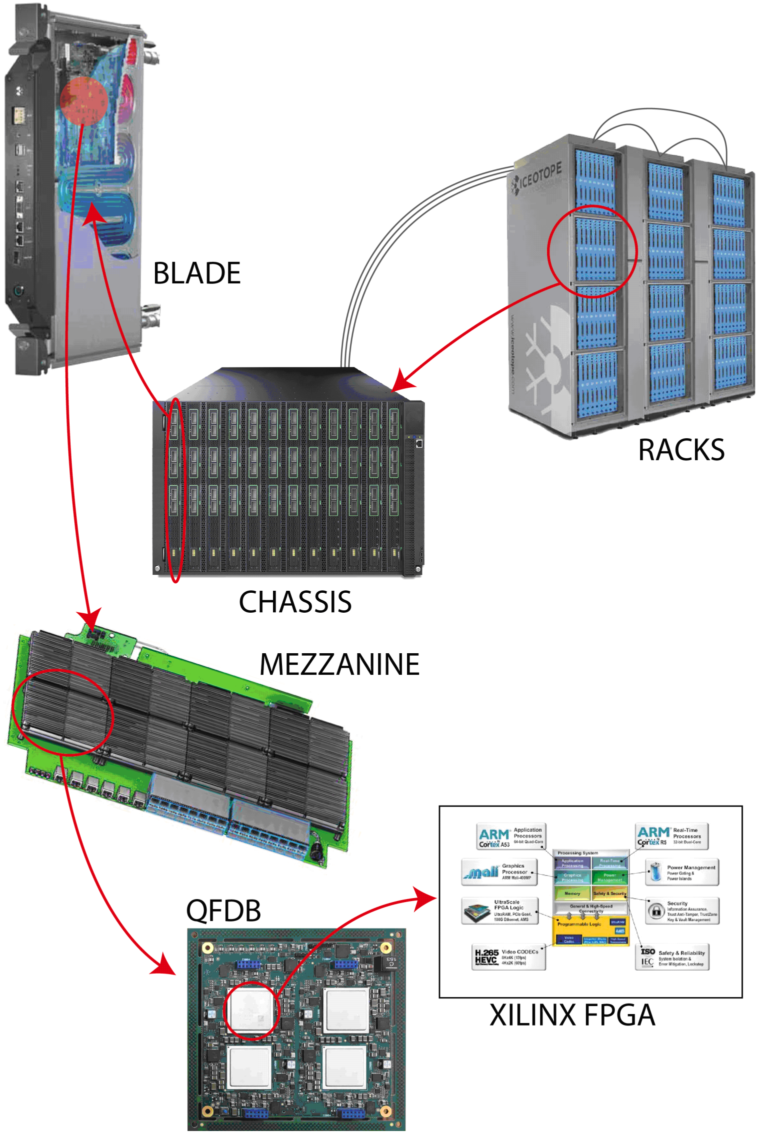
On the Effects of Data-aware Allocation on Fully Distributed Storage Systems for Exascale

JA. Pascual, C Concatto, J Lant, and **J Navaridas**
School of Computer Science
The University of Manchester

Horizon 2020

The **ExaNeSt** HPC project: Storage, Interconnect, Cooling

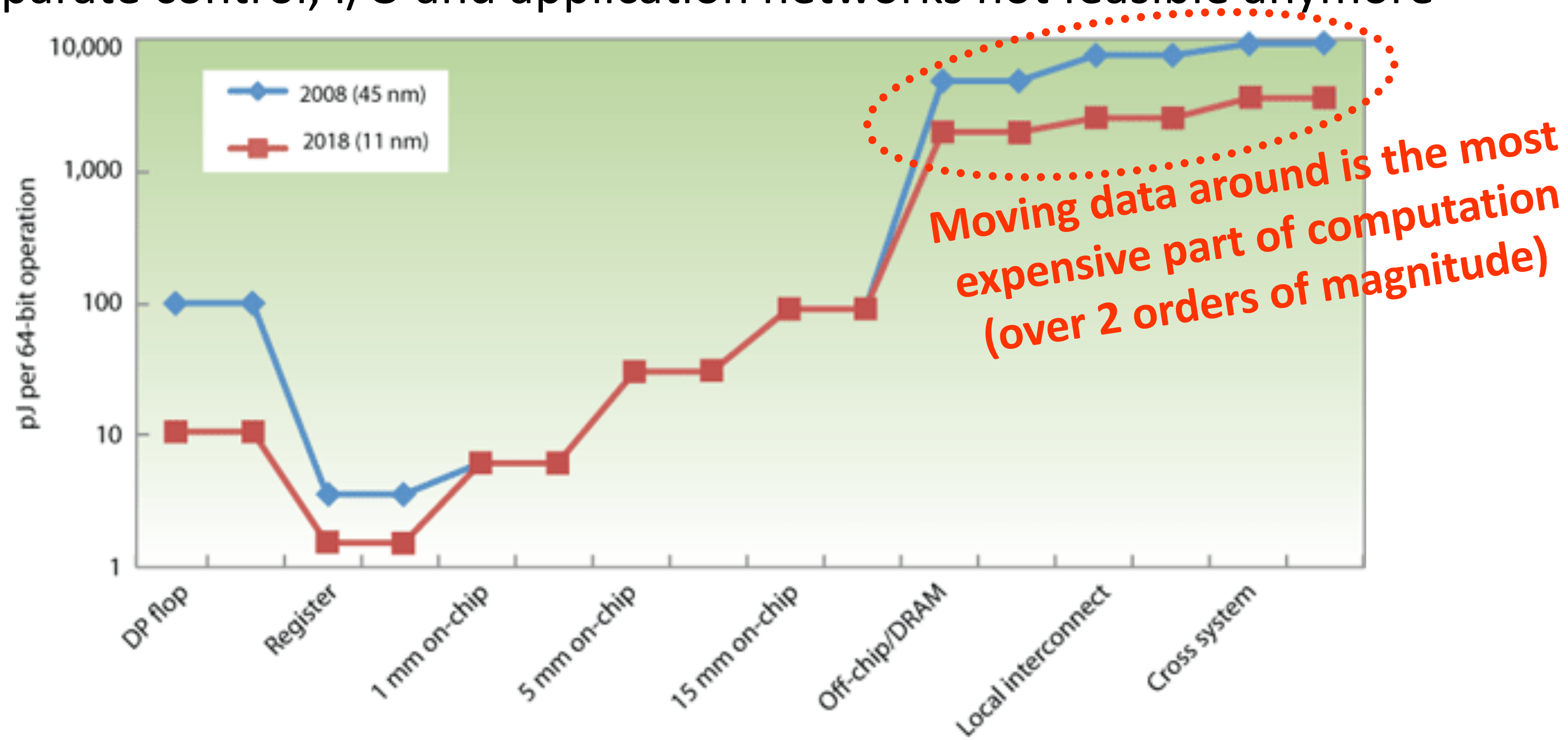
- **Storage**: fast, distributed in-node non-volatile memory
- **Interconnect**: low-latency, unified compute & storage traffic
- **Packaging**: advanced, liquid-cooled
- **App's**: real, scientific and datacenter
- **Prototype**: 1000+ ARM cores & 200+ FPGAs



ExaNeSt project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 671553

Low-energy is essential to reach Exascale

- **Mobile processors + reconfigurable logic (FPGA fabric)**
  - Up to one order of magnitude more Flops/Watt than HPC processors
- **Near Data computation** to reduce data movement
  - Distributed storage system with one NVM per node
- Unifying **networking infrastructure** is also necessary
  - The interconnect can consume up to 25% of the total power
  - Separate control, I/O and application networks not feasible anymore



Objectives of this paper

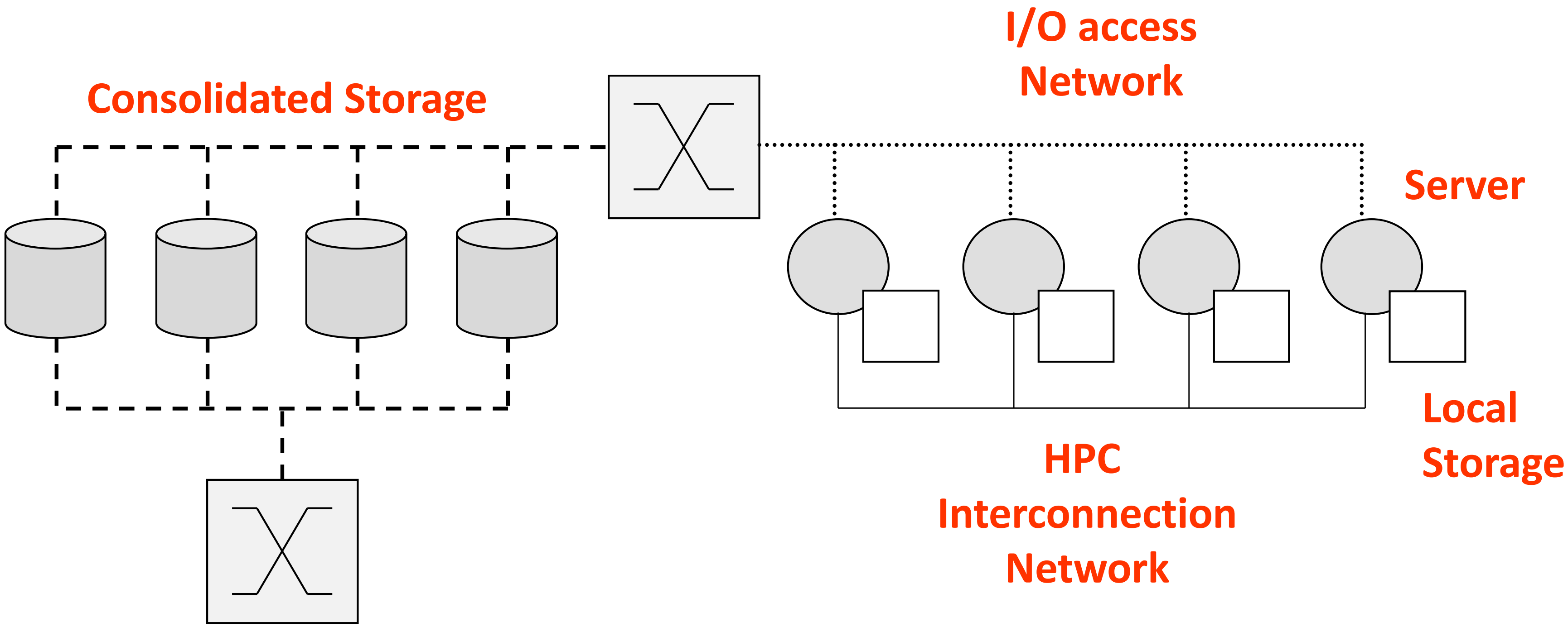
We want to understand:

- Effects of merging inter-processor and storage traffic
  - To what extent do they interfere with each other?
  - How this affects performance?

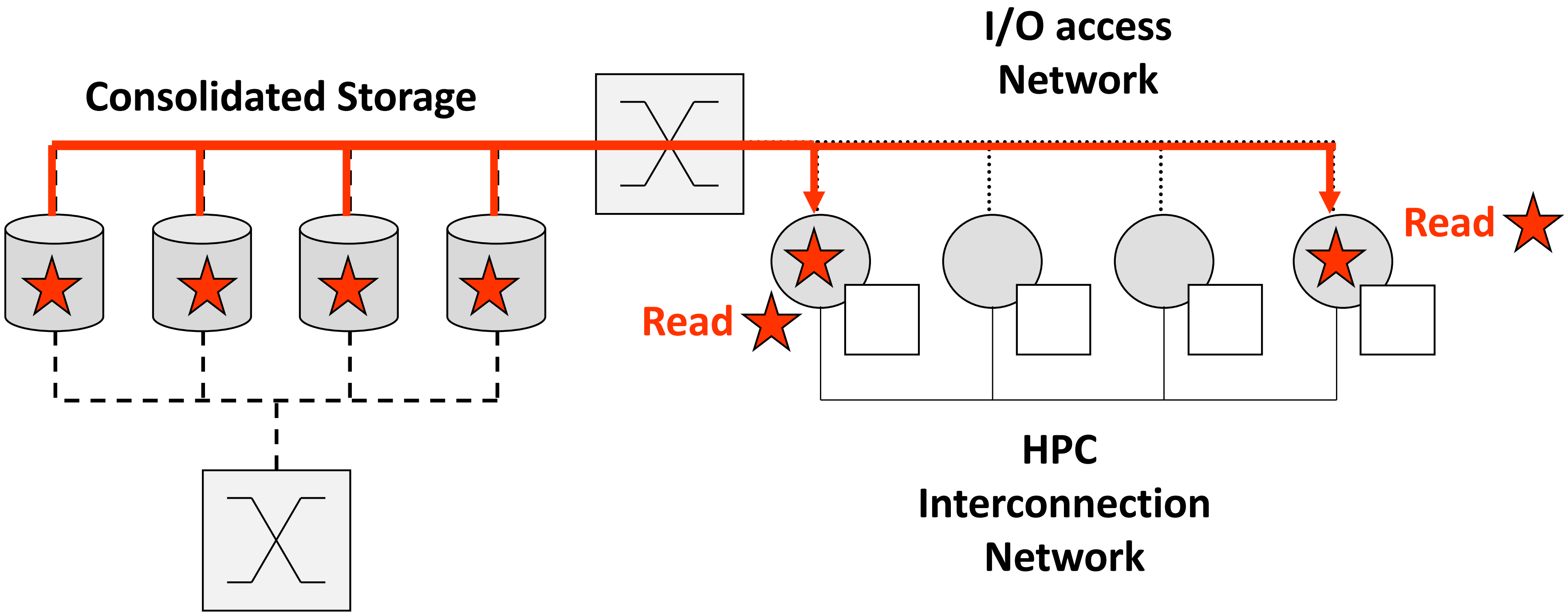
- Effects of Locality
  - How sensitive to temporal locality of data are applications?
  - What about spatial locality?

- Based on those:
  - Can resource allocation (tasks and Data) help?

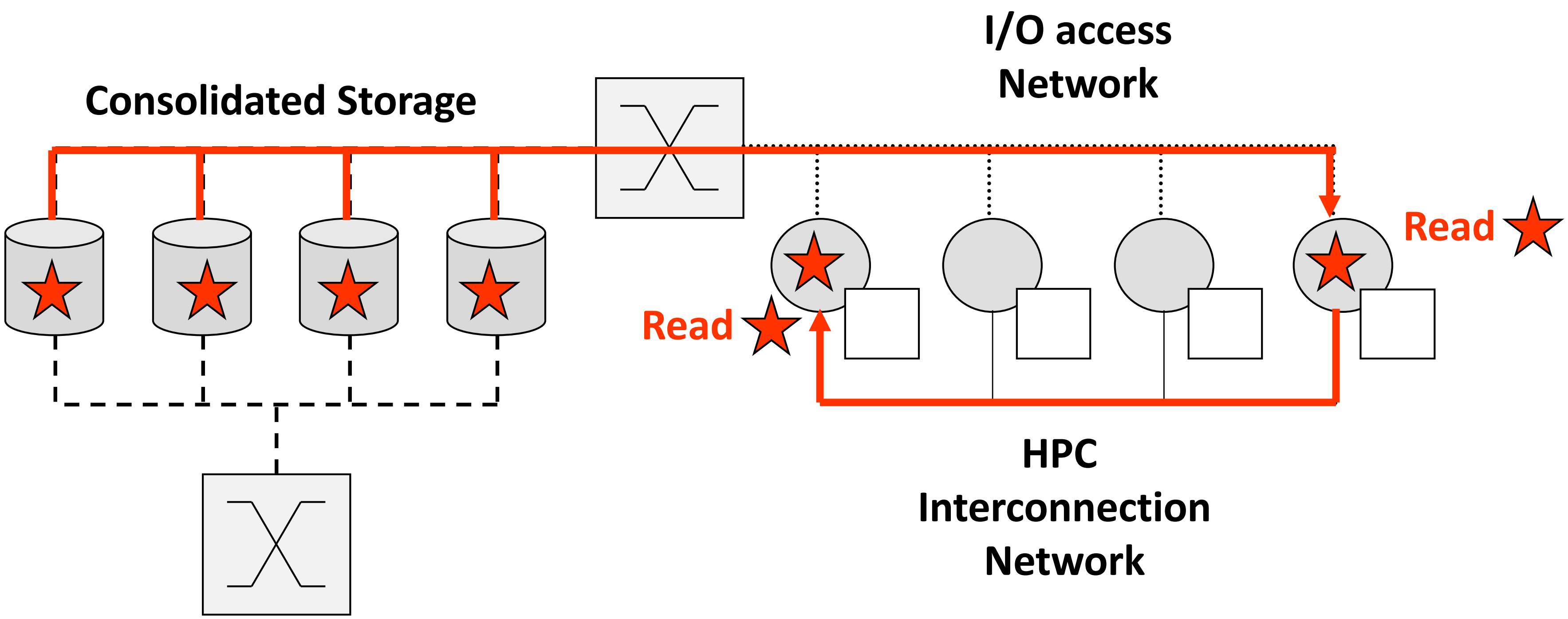
System Architecture



Traditional SAN-based I/O Infrastructure

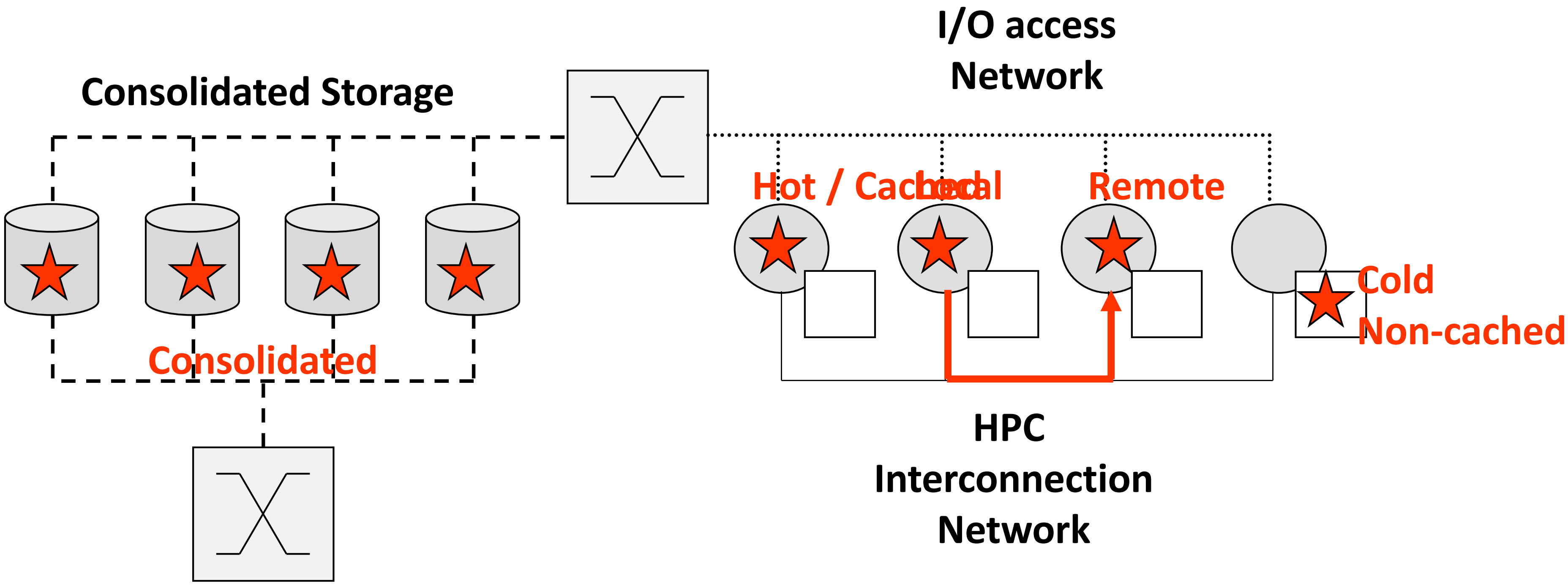


Parallel I/O Infrastructure

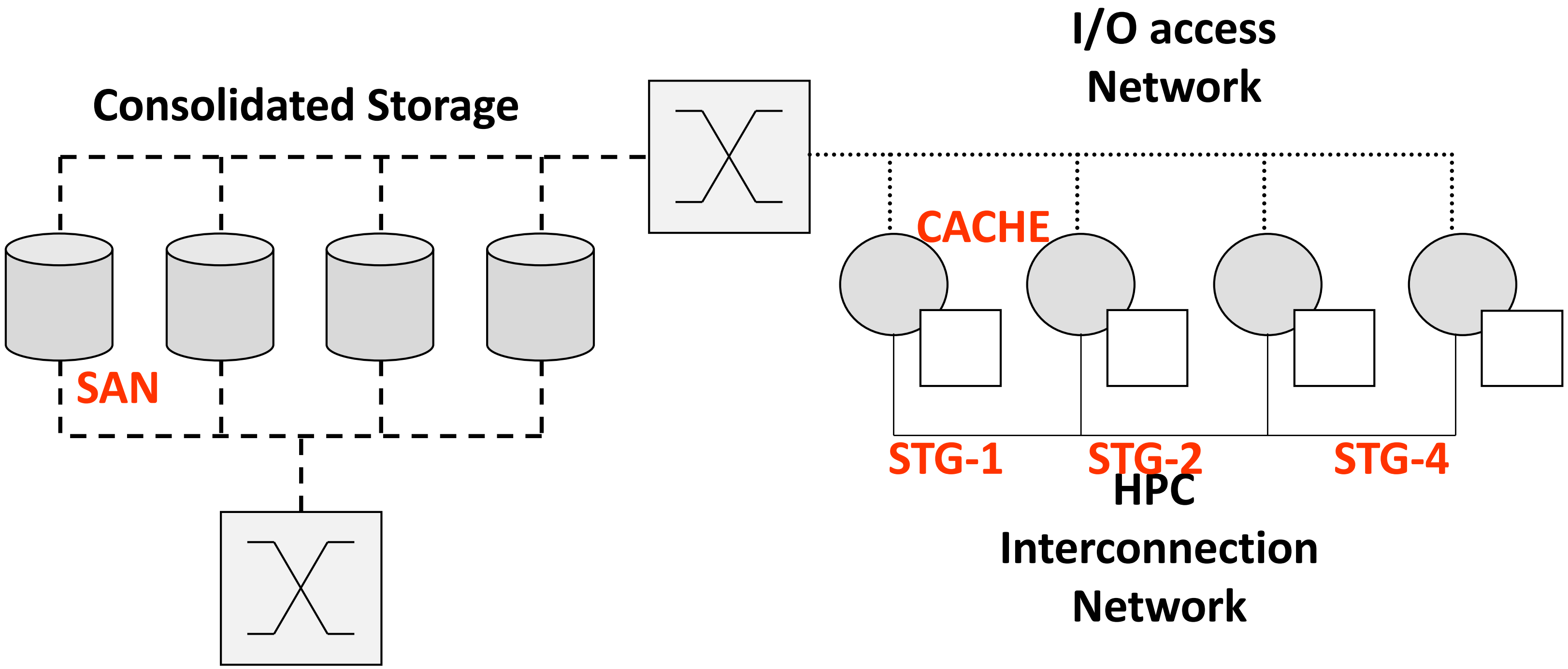


In ExaNeSt this is handled transparently by BeeGFS
See https://www.beegfs.io/

Nomenclature



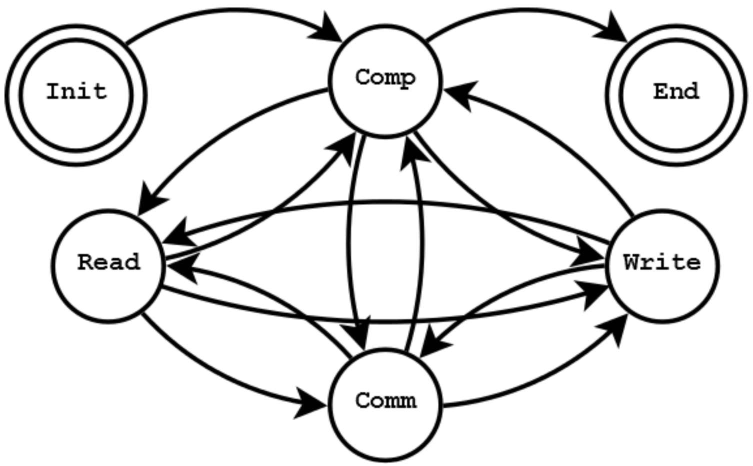
Allocation policies under evaluation



Experimental Set-up

- Simulation-based evaluation using INRFlow
  - Fat-tree and torus-like topologies
  - Single-application experiments with 64 nodes
  - Multi-application experiments with 512 nodes and 4×128-node apps

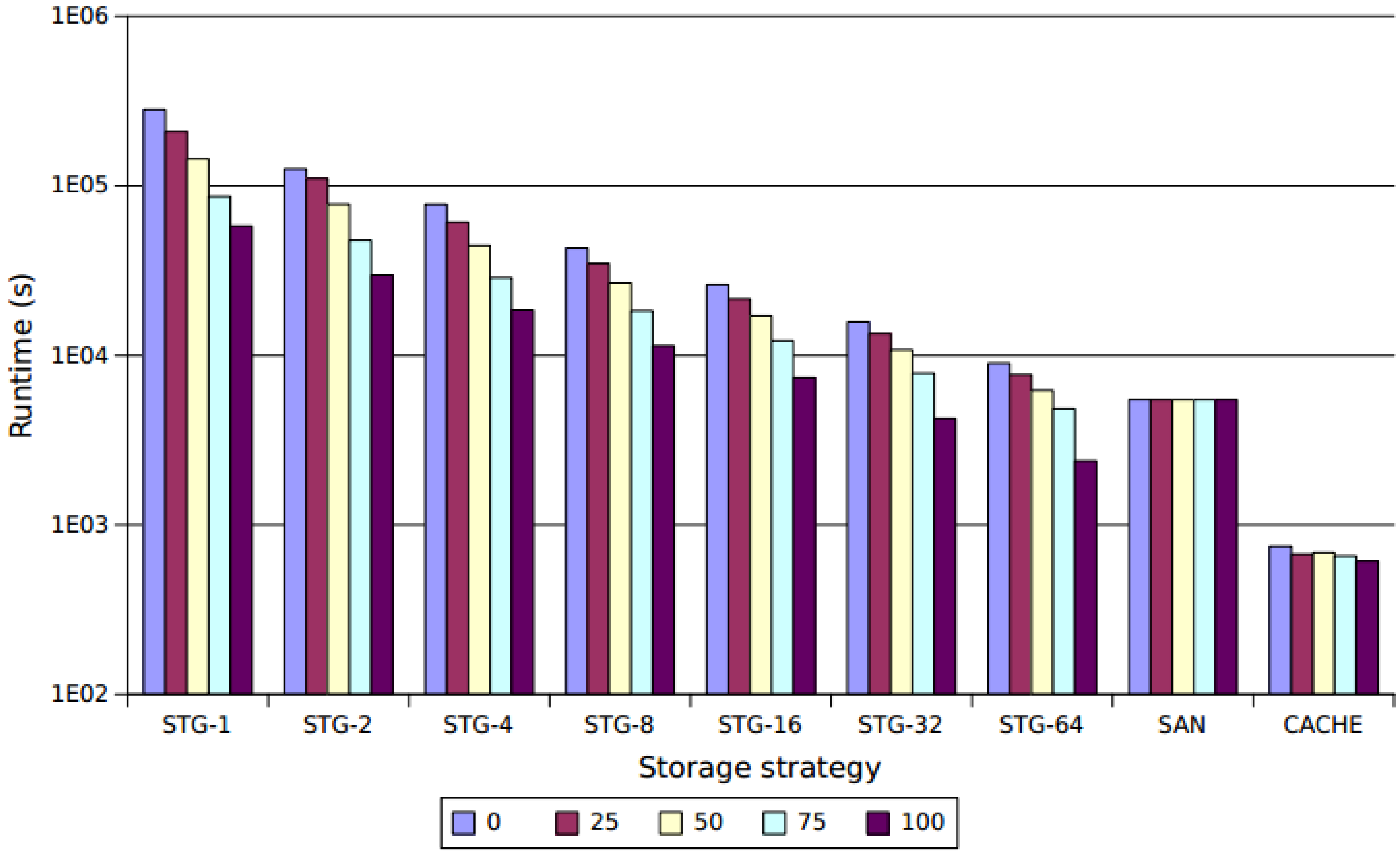
- Data-intensive Markov-based application model
  - 75% I/O
  - 12.5% communication
  - 12.5% computation



- Several data allocation strategies
  - Data always in local **CACHE** (best case)
  - Data always from **SAN** (64 I/O Nodes - 40Gbps network)
  - Data stored in ***k*** local NVMs - random access (**STG-*k***)

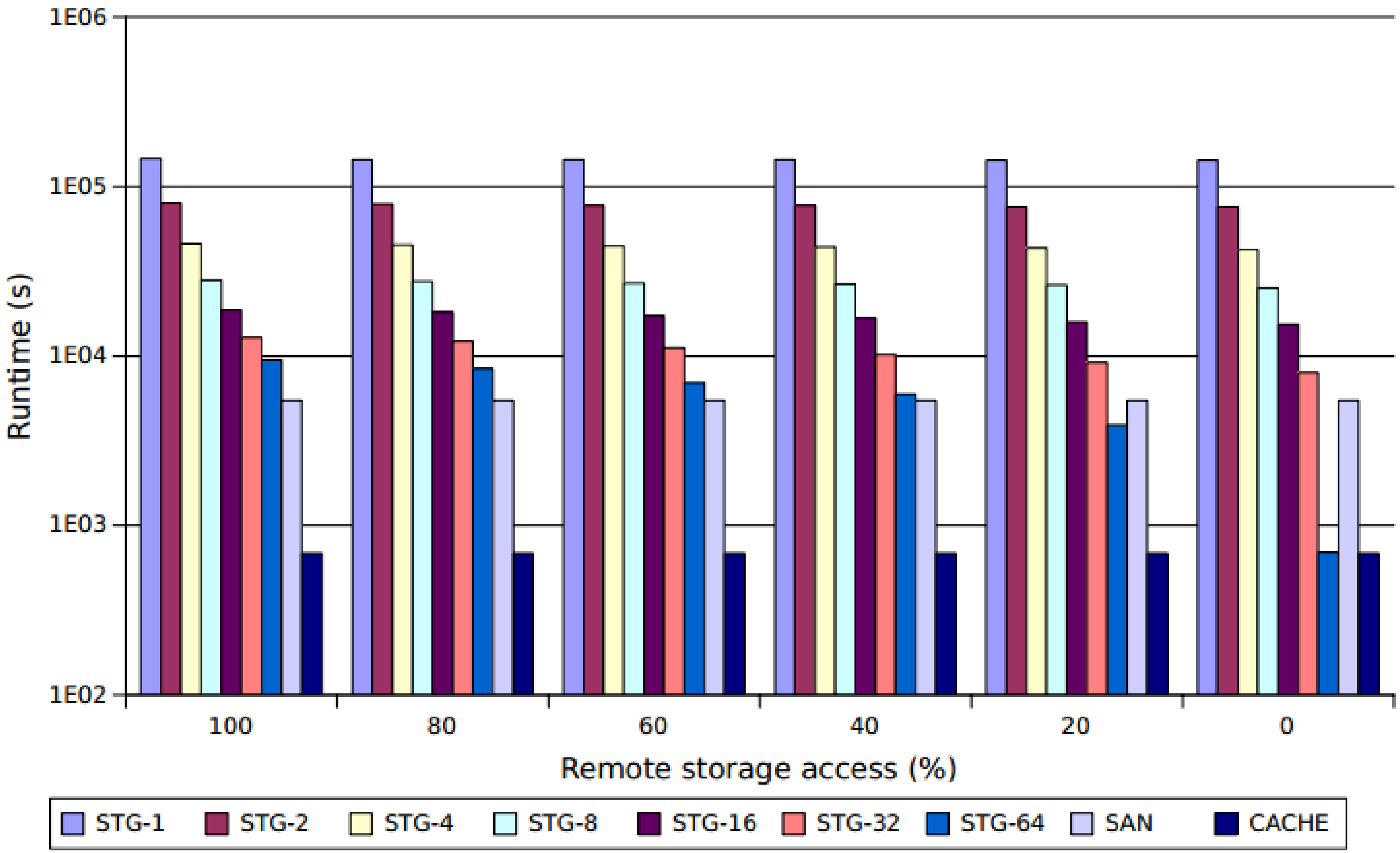
Single Application – Temporal locality

- Spreading Data across many local NVMs can be very useful
  - Nearly linear scaling of performance for random access
- Exploiting temporal locality is crucial
  - Up to 5× slower with *Cold* data



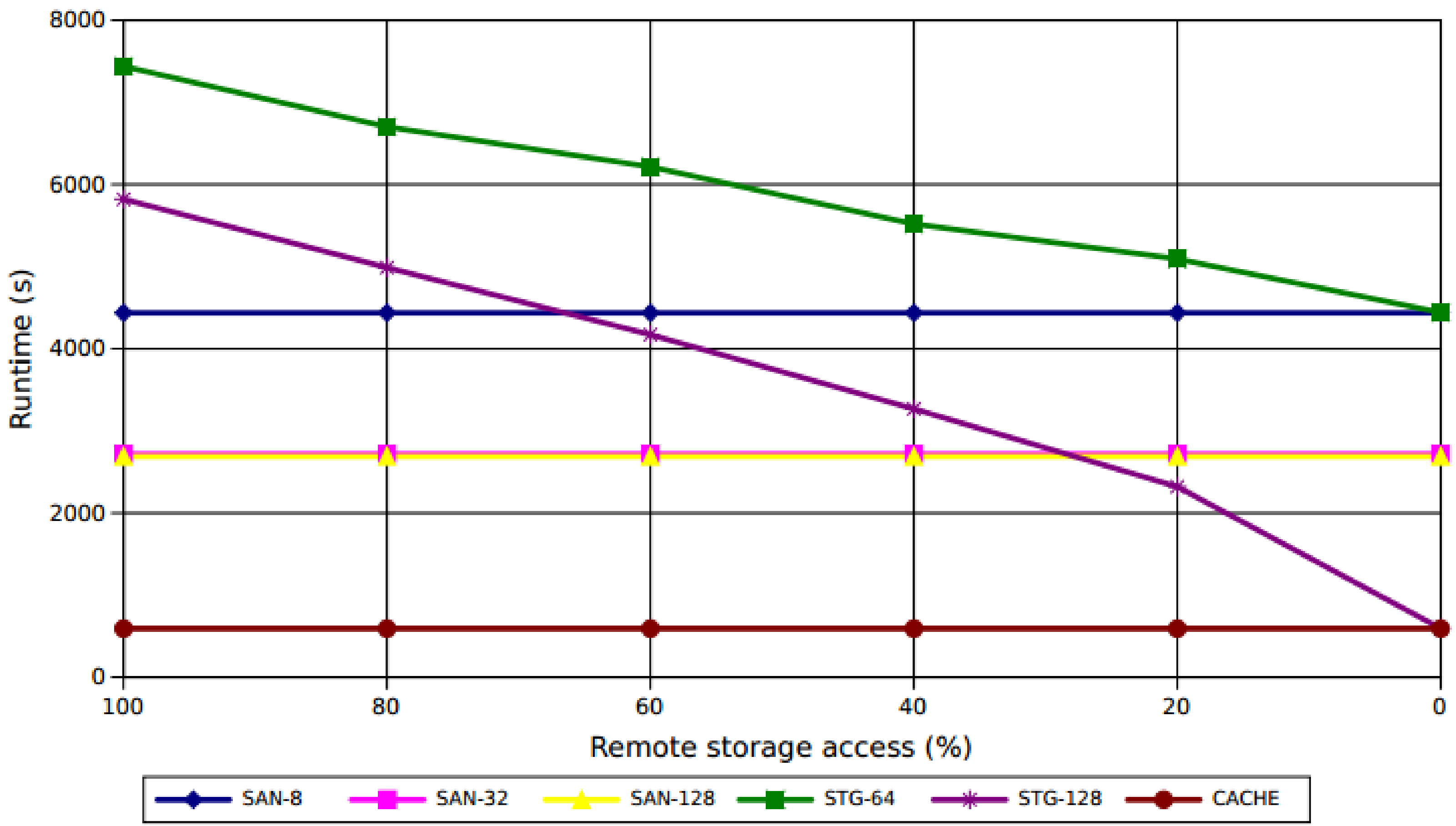
Single Application – Spatial Locality

- Large numbers of NVMs benefit greatly from locality
  - Over one order of magnitude slower with poor locality
- Exploiting locality (~60%) can outperform the *big* I/O SAN



Multi-application – Spatial Locality

- Similar results for multi-application scenarios
  - Using more NVMs can substantially improve the performance
  - The more NVMs, the more potential benefit from exploiting locality
- SAN infrastructure needs to be overprovisioned



Conclusions and future work

- Centralized I/O architectures are prohibited in terms of energy
  - Need to rely on distributed approaches that minimise data movement
- Spatial and temporal locality have a great impact on Storage
  - There is great potential for Data and Tasks allocation policies

- Further work arising from this study
  - Develop Data- and Task-aware strategies for resource allocation
  - Develop flow-allocation schemes to reduce traffic interference
  - Continue work on the best topological arrangement
  - Develop QoS and congestion control mechanisms
  - Keep refining simulation models for applications and systems