A Foundation for Automated Placement of Data

Douglass Otstott, Sean Williams, Latchesar Ionkov, Michael Lang, Ming Zhao

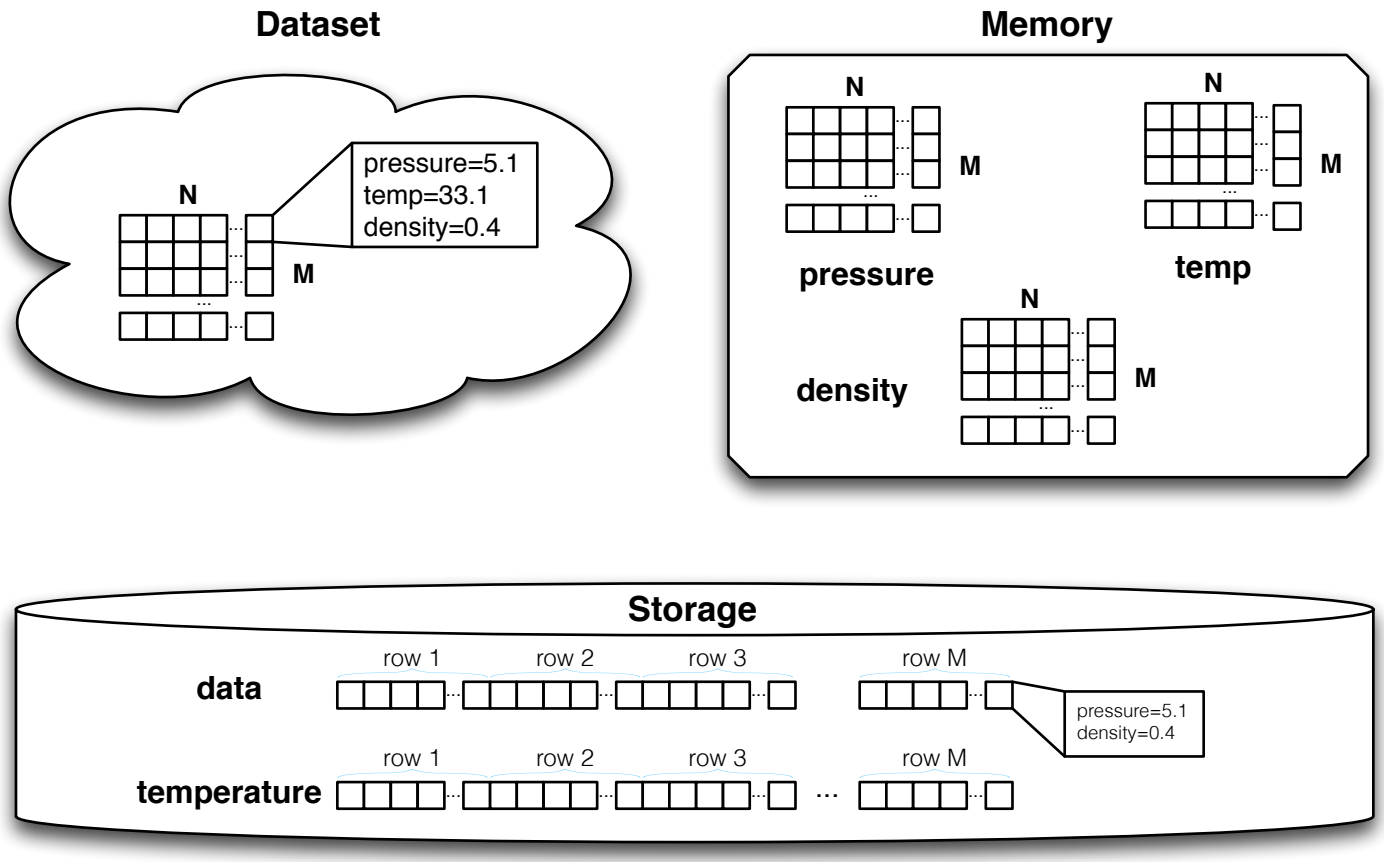
LA-UR-17-22686

Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Memory and Storage are Converging

- Persistent storage on the memory bus (NVDIMMs)
- Remote memory (GenZ)
- Which memory bus? (DRAM, HBM, GPU memory, …)

Data Layouts are Different



Data Sharing

- With less distinction — more confusion
- With more complicated workloads there are a lot of options
- In situ, in transit, …

- No generic way for sharing data in memory between applications
  - ad-hoc
  - in-memory file system

- What data format?
  - data producer
  - data consumer

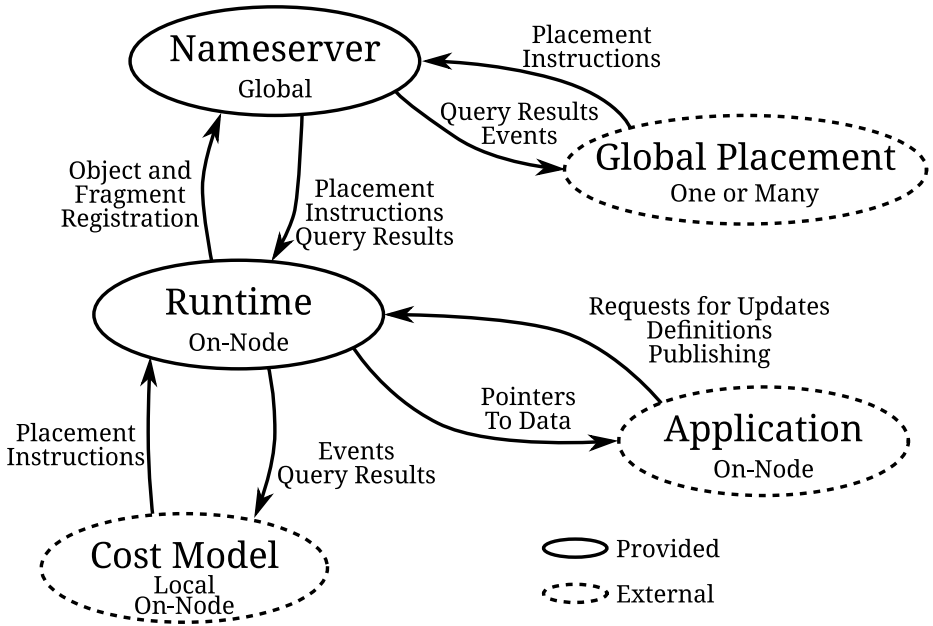
Need for Data Management Service

- Handles all data that application shares
- Moves data between the many memory and storage layers
- Allows data layout transformations

- This work
  - describes the foundations for building such service
  - allows data movement and transformation
  - doesn’t include the support for global data optimizations

Components

- Name server
  - handles metadata
  - global
- Runtime
  - runs on every node
  - handles local data
  - talks to runtimes on other nodes
- Global/Local placement services (not included)
  - optimize data locality and format
- Application (not included)



Data Model

- Dataset
  - types
    - primitive types (integer, floating point, string)
    - structs
    - (multidimensional) arrays
  - variables

- Fragments
  - subsets of a dataset
  - types - based on dataset types
  - variables - based on dataset variables

- Versions
  - provide consistent view of distributed dataset

Declarative Data Language & Transformations

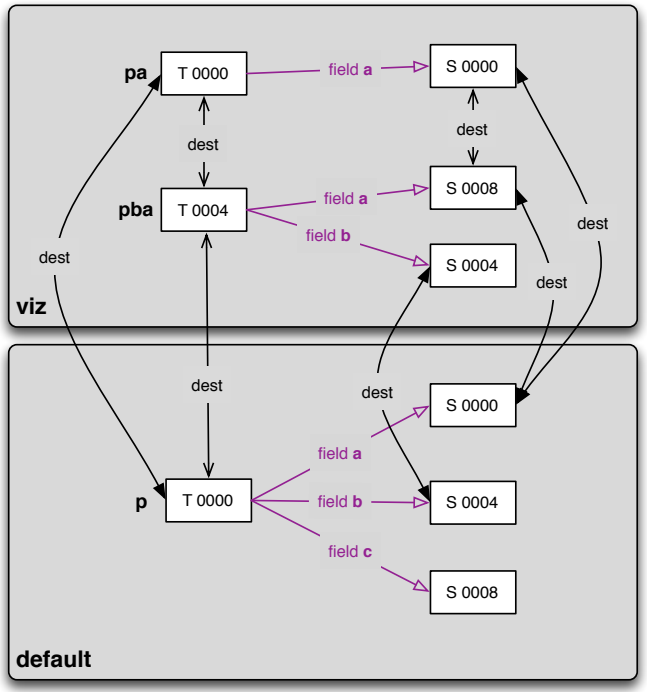
- For the user: define the abstract dataset and subsets

```
fragment dataset {
    var p struct {
        a, b, c float64
    }
}

fragment default {
    var p = p
}

fragment viz {
    var pa { a } = p
    var pba { b, a } = p
}
```

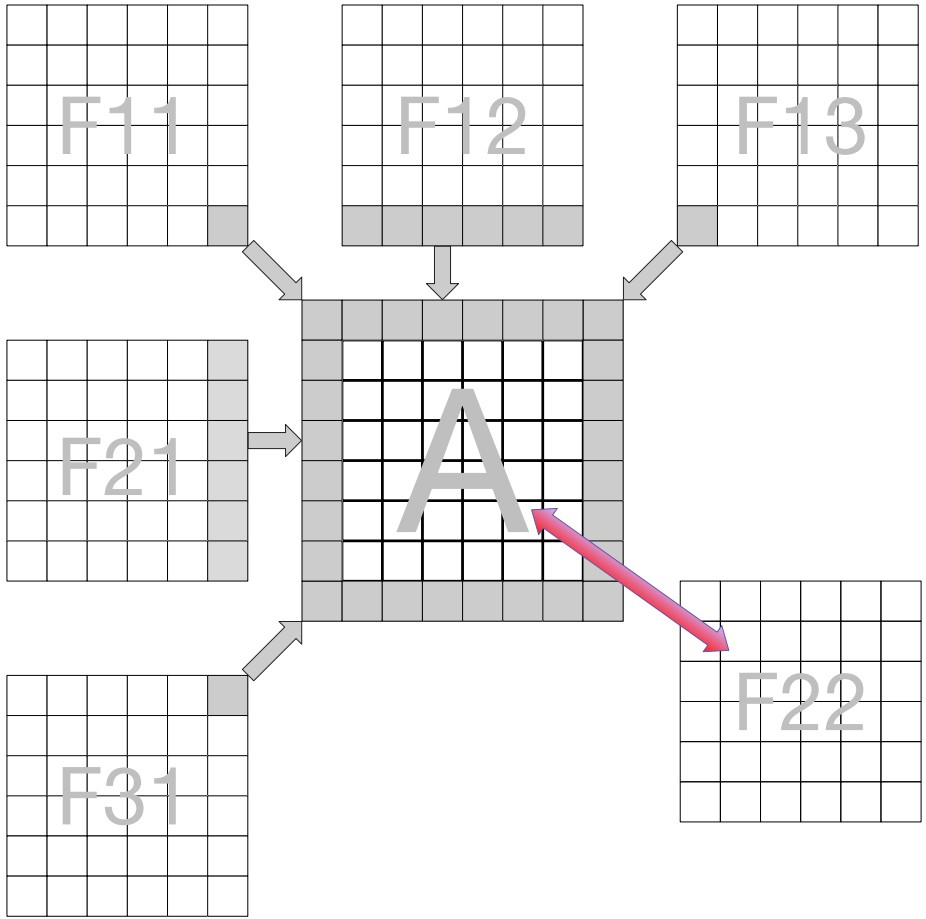
- For the computers: transformation rules that convert data between dataset and subsets



- API
  - create object
    - name
    - dataset description
  - attach fragment
    - dataset name
    - fragment description
    - version
  - publish fragment
    - data pointer
    - version

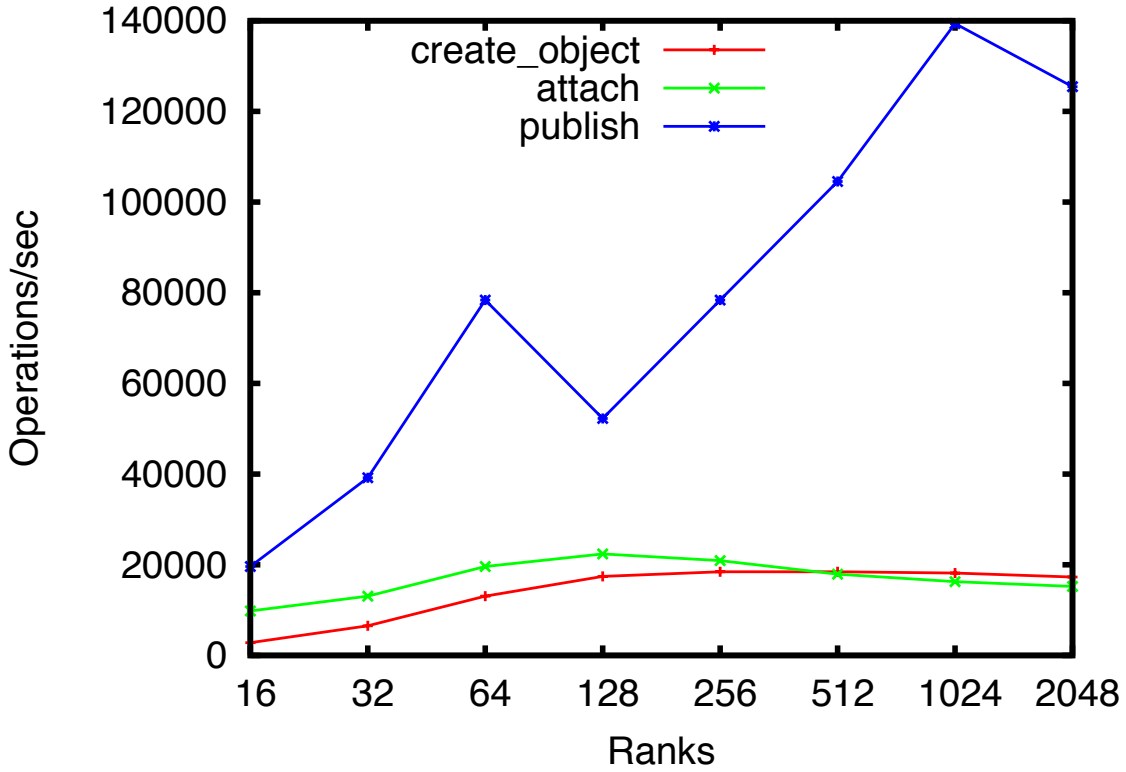
- Operations
  - object registered in the name server
  - runtime
    - finds the locations of necessary fragments that contain the relevant data and version
    - brings the data and transforms it to the required format
  - runtime
    - registers the fragment version in the name server
    - keeps copy of the data in memory or local storage

- Can be used for communication between ranks
- Fragment can have read-only and read-write parts of complex geometry



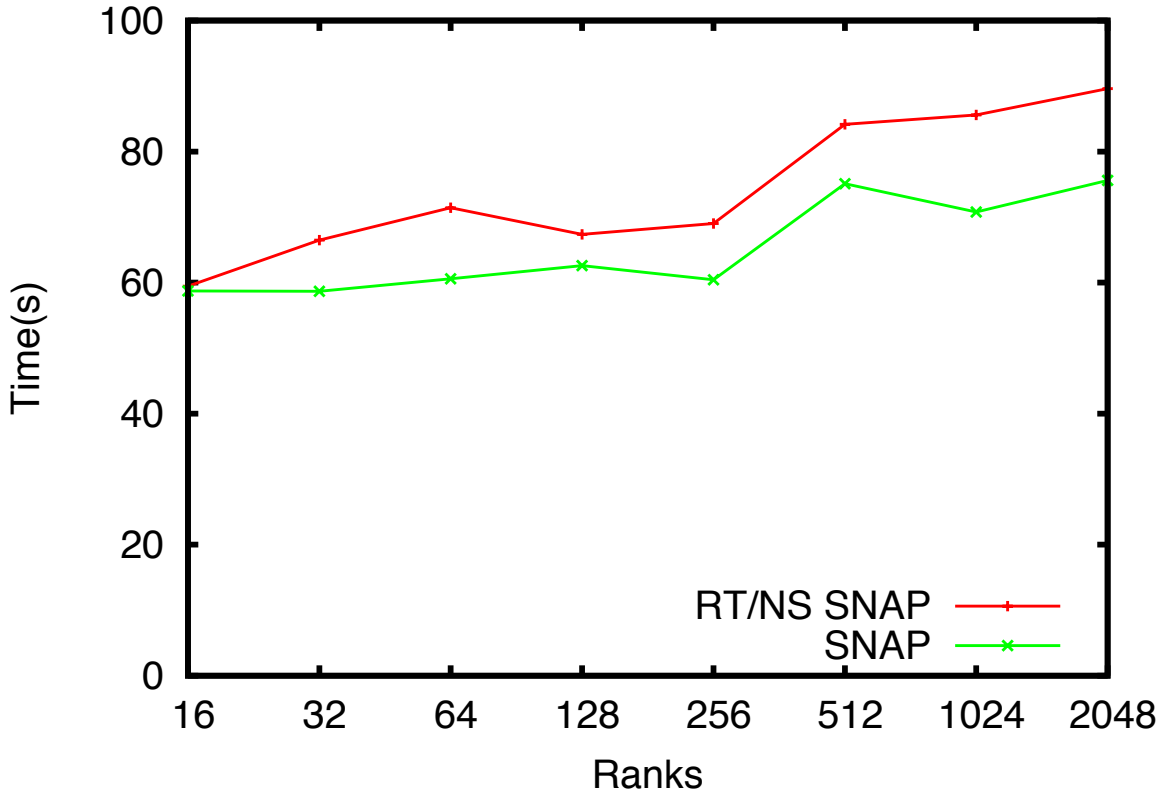
Results

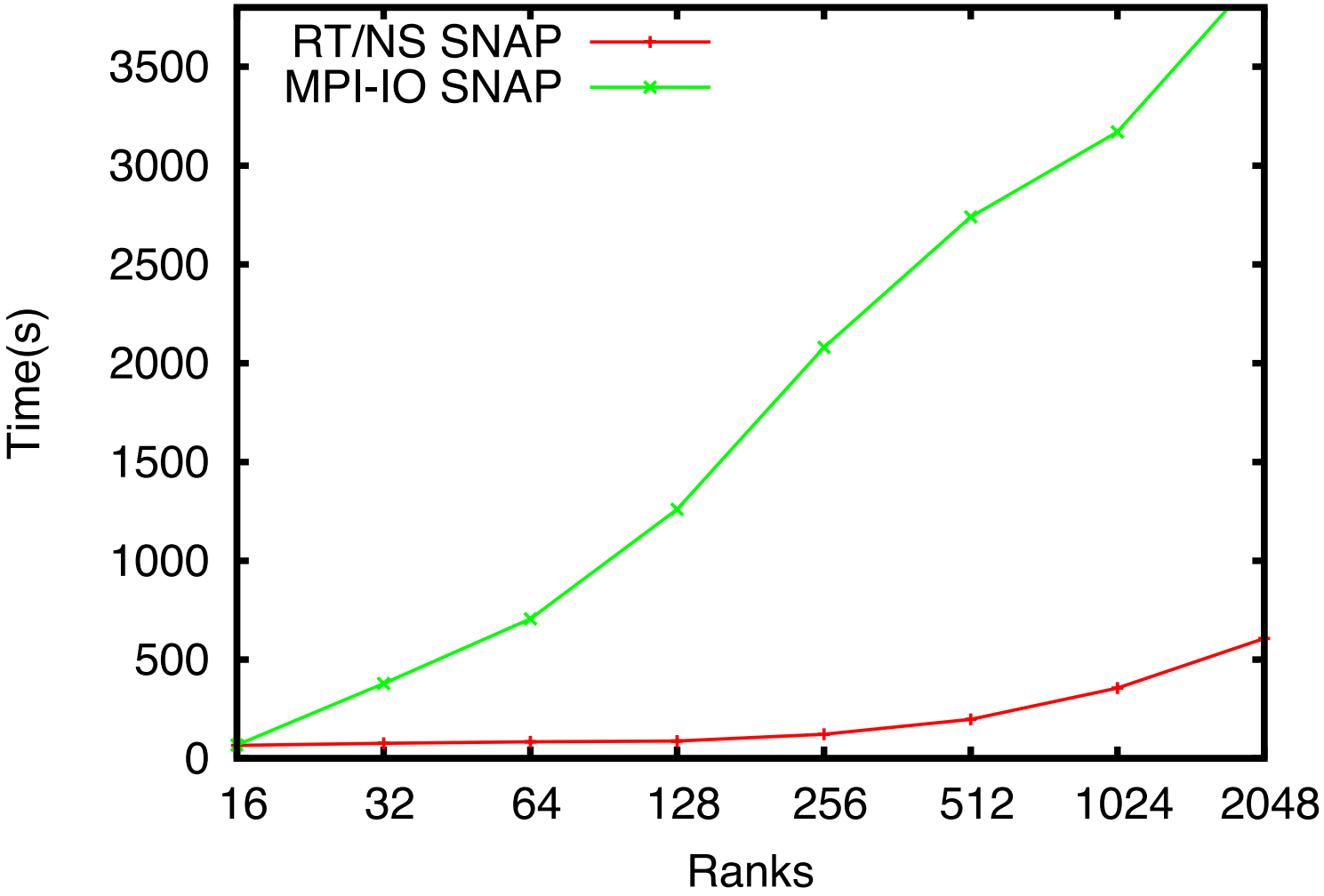
- Synthetic benchmark
- Evaluates the overhead of the operations
- Single name server
- 16 ranks per node

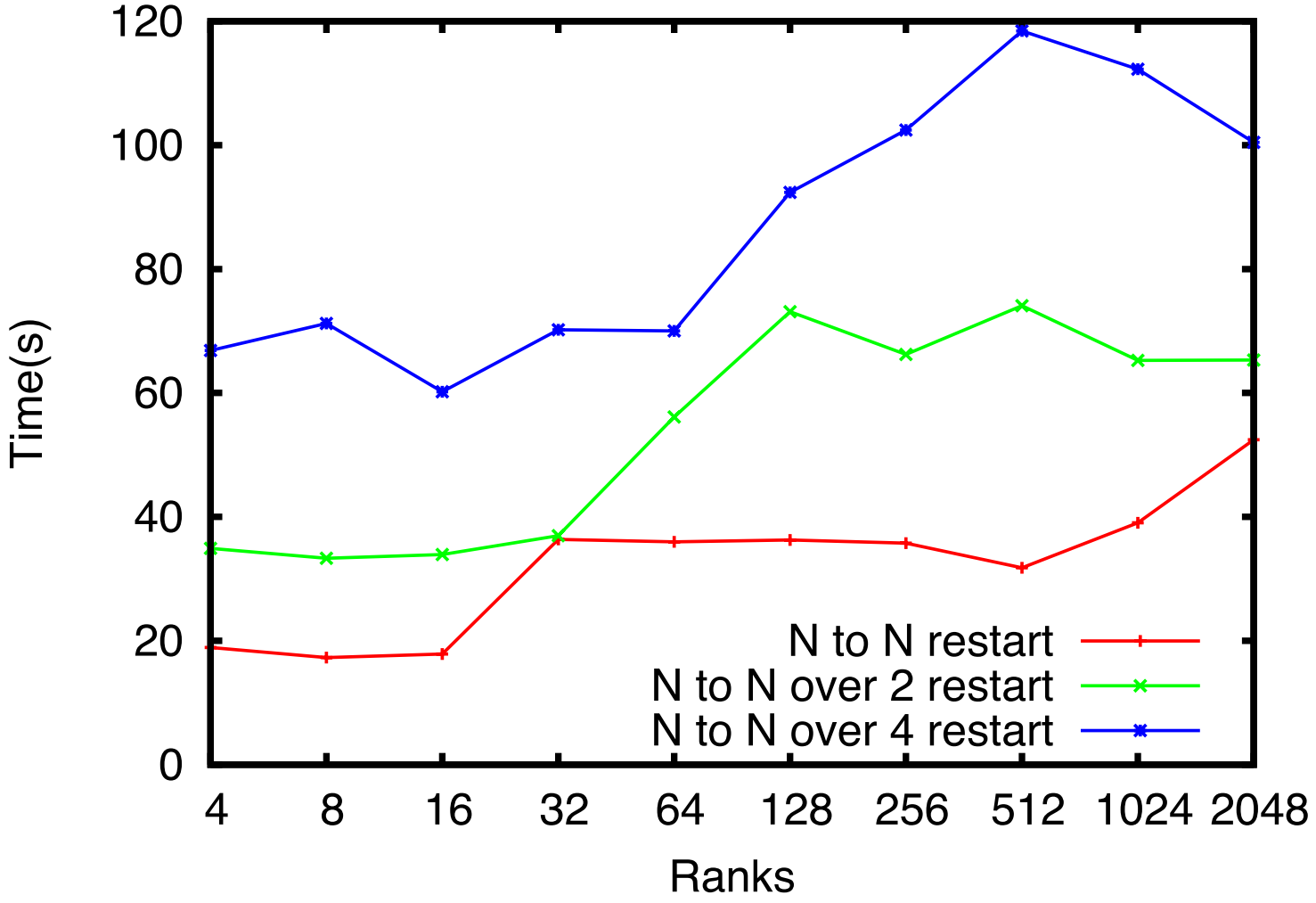


Results: SNAP checkpoint

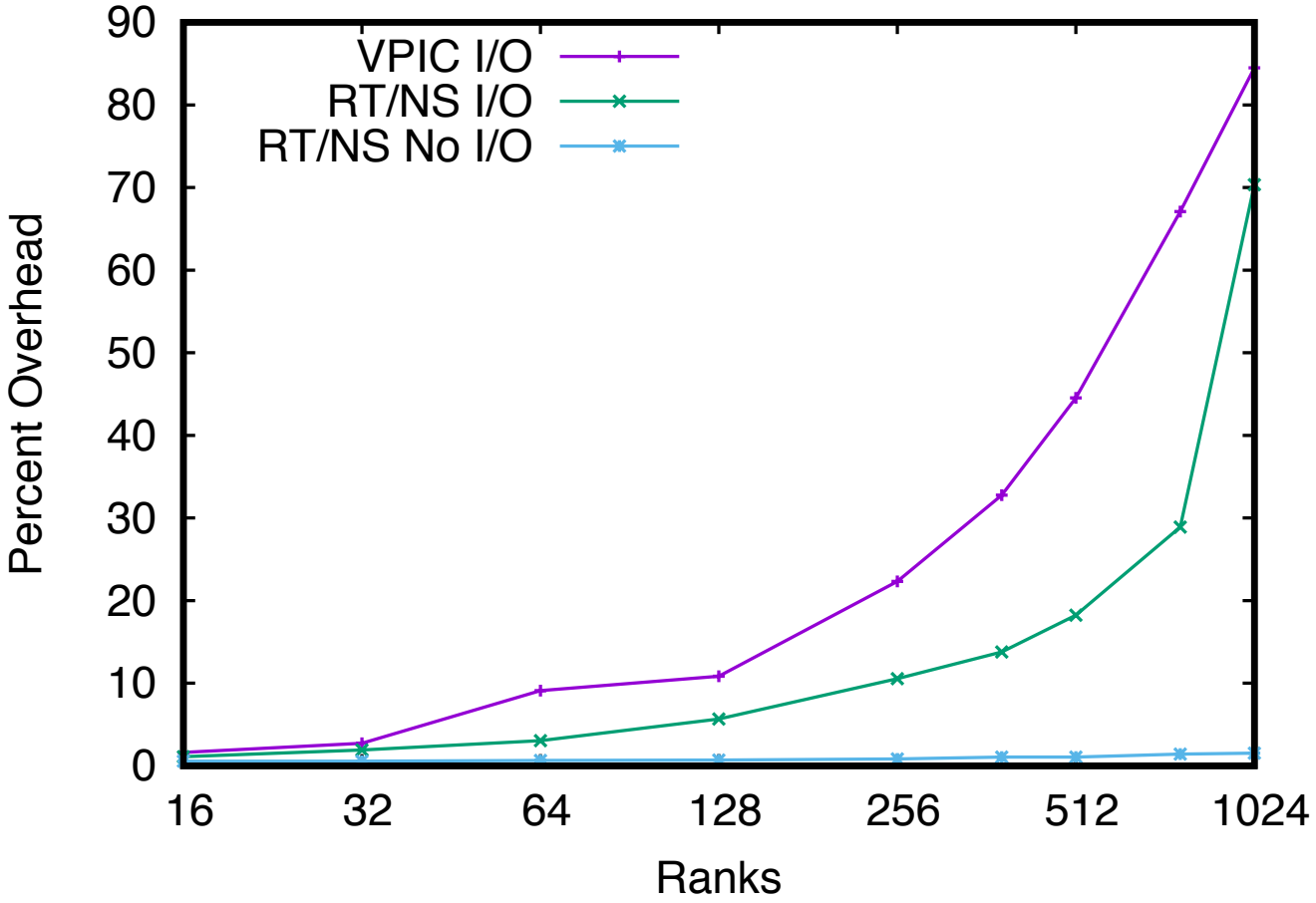
- Original SNAP (no checkpoints) vs. adding the checkpoint code
- Evaluate the overhead




RT/NS SNAP
MPI-IO SNAP
Time(s)
0
500
1000
1500
2000
2500
3000
3500
16
32
64
128
256
512
1024
2048
Ranks


120
100
80
60
40
20
0
Time(s)
4
8
16
32
64
128
256
512
1024
2048
Ranks
N to N restart
N to N over 2 restart
N to N over 4 restart

Results: VPIC



Conclusions

- Scalable data service
- Easy to use API

- Future
  - Integration with data placement services
  - Additional applications (E3SM)
  - Scalable name server