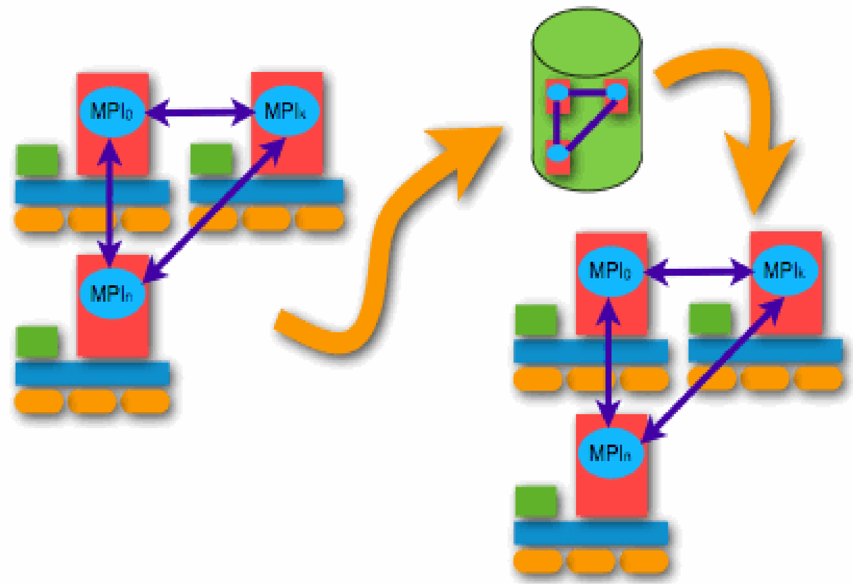
VELOC: Deep Learning Support

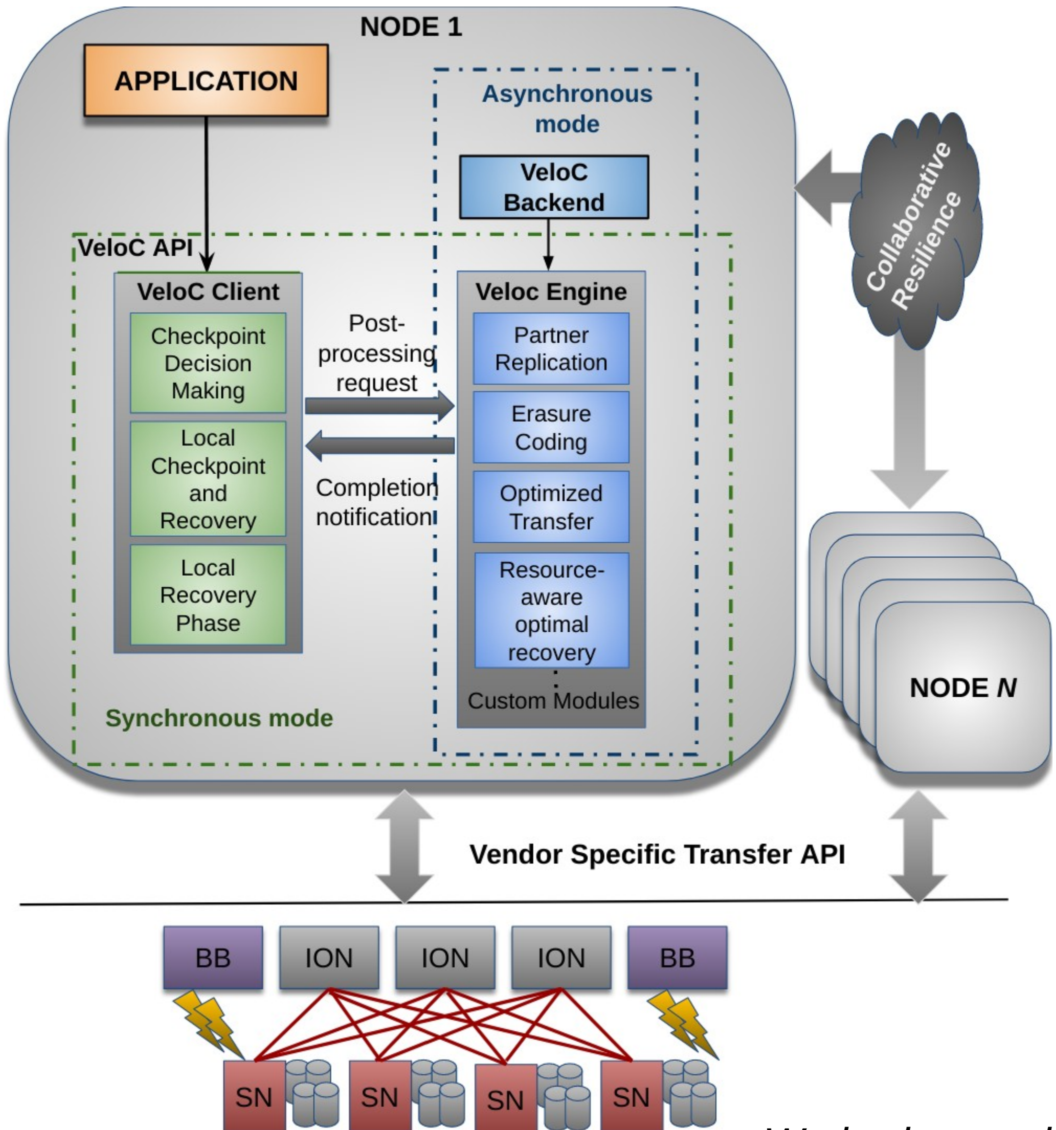
Bogdan Nicolae
Mathematics and Computer Science
Argonne National Laboratory

State Preservation in HPC

- More than checkpointing: manages relationships between intermediate checkpoints
- Defensive:
  - Fault tolerance based on checkpoint-restart
- Administrative:
  - Suspend-resume (e.g. make room for higher priority jobs)
  - Migration
  - Debugging
- Productive:
  - Share and reuse between simulations and analytics
  - Revisit previous intermediate datasets (e.g. adjoint computations)
  - Provenance tracking



VeloC Architecture

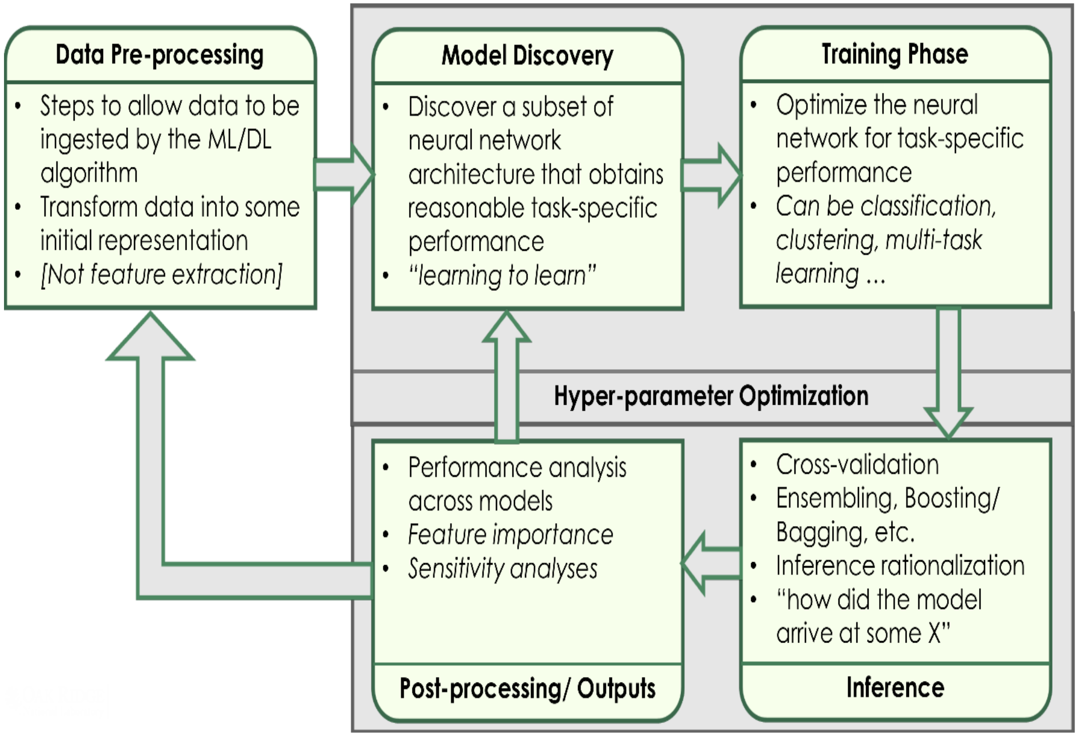


- High Performance and Scalability
- Hides complexity of interaction with deep storage stacks
- Configurable resilience strategy:
  - L1: Local write
  - L2: Partner replication, XOR encoding, RS encoding
  - L3: Optimized transfer to external storage
- Configurable mode of operation:
  - Synchronous mode: resilience engine runs in application process
  - Asynchronous mode: resilience engine in separate backend process (VeloC does not die if app dies due to software failures)
- Easily extensible:
  - Custom modules can be added for additional post-processing in the engine (e.g. compression)

We had a tutorial on Tue!
Web: ***https://veloc.readthedocs.io***

EXASCALE COMPUTING PROJECT

Use Cases Beyond Resilience: Deep Learning

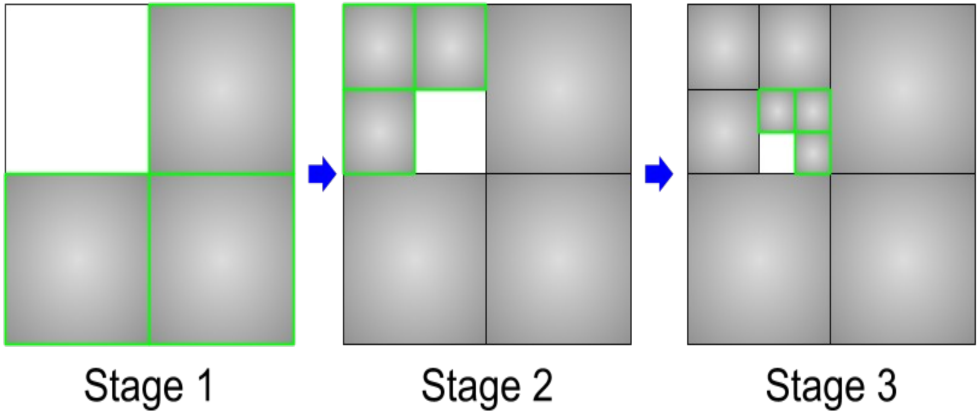


- Defensive + Administrative:
  - Save intermediate models during exploration/training to enable restart + suspend-resume, persistence between jobs
- Productive:
  - Organization of training data (cache & reuse after pre-processing, partitioning, shuffling)
  - Hyper-parameter search using evolutionary/population based techniques that revisit intermediate models
  - Knowledge transfer
  - Swapping for large models that cause OOM
  - Introspection: Study intermediate states to understand evolution of weights

- Key challenge: need to try many possible network configurations, each defined by many possible parameters
- Becomes a big data problem
- Example: CANDLE (more than 650GB/s on the Summit pre-Exascale machine)

ECP CANDLE: Data Analysis Workflow

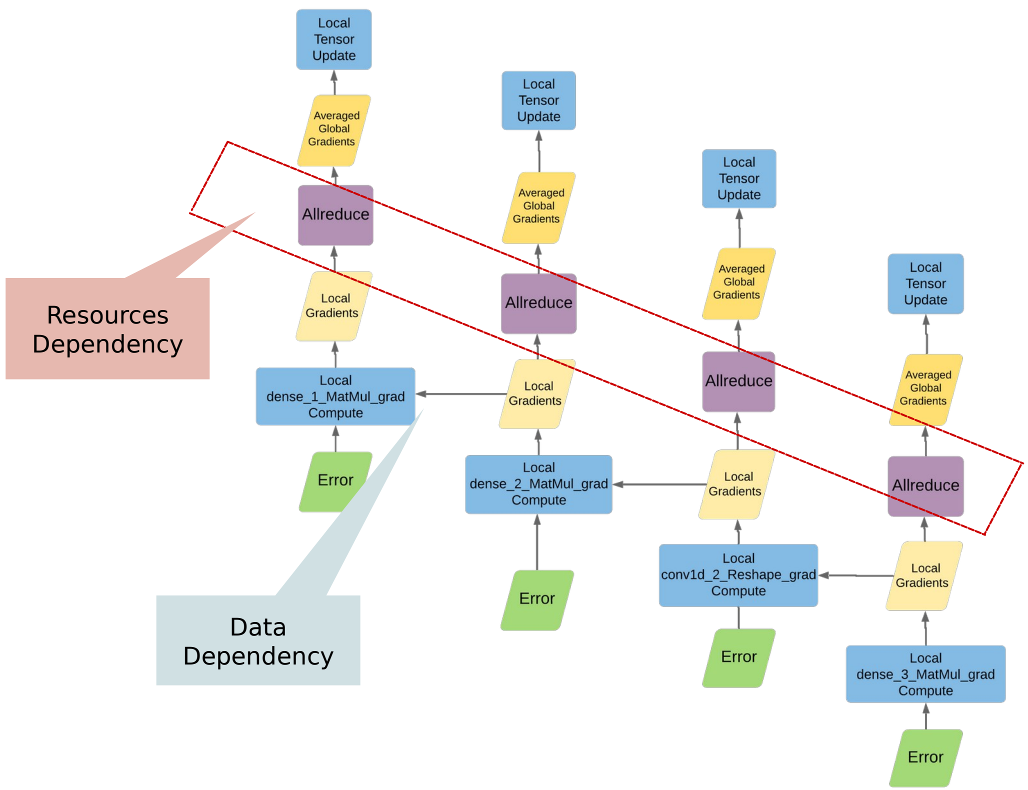
- Split up the training data into subsets, iteratively train on most remaining subsets.
- Weight sharing from one subset to the next (incremental learning)
- Allows for investigations into data quality and learning patterns
- Could also boost performance by preventing overloading data ingest limits
- Recursive calls define the datasets for training
- Runs at large scale on Summit, ramp-up/down
- Many models are written and read

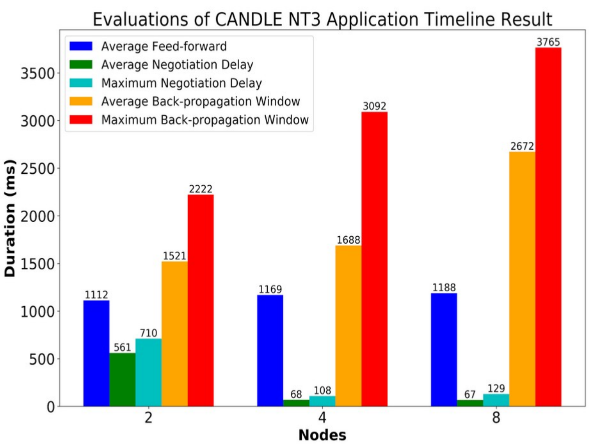


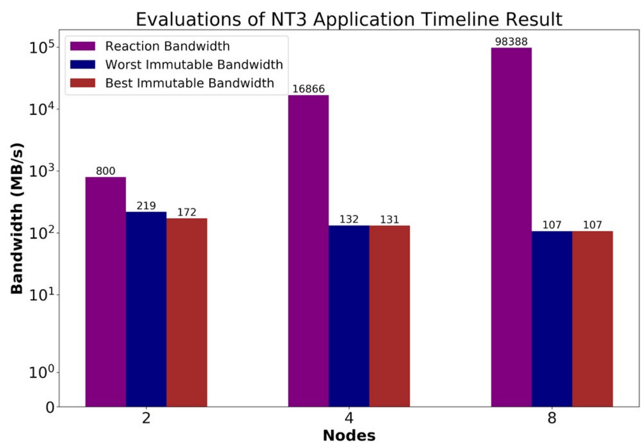

```
run_stage(int N, int S, string this, int stage,
          void block, string plan_id) {
  void parent = run_single(this, stage, block, plan_id);
  if (stage < S) {
    foreach id_child in [1:N] {
      run_stage(N, S, this+"."+id_child, stage+1, parent,
                plan_id, db_file, runtype);
    }}
}

run_single(string node, int stage, void block) {
  json_fragment = make_json_fragment(node, stage);
  json = "{\"node\": \"%s\", %s}" % (node,
json_fragment);
  block => obj(json, node);
}
```

Check out ECP CANDLE slides for more details: http://tiny.cc/zwpojz

Study: CANDLE NT3 Benchmark







- Fine-grain parallelism of back-propagation for data-parallel training
  - Reference implementation: Horovod
  - Resource dependency: One all-reduce at a time
  - Data dependency: Wait for lower layers
- Study of fine-grain parallelism
  - Introduced metrics and scripts to process Horovod timeline
  - Results show significant opportunity to embed asynchronous operations into the pipeline (e.g., after local tensor update)
  - Key takeaway: We can help with understanding bottlenecks and system-level aspects related to data-parallel training
- Checkpointing of models:
  - We have developed efficient async techniques based on VeloC

Jiali Li, Bogdan Nicolae, Justin Wozniak, and George Bosilca. Understanding scalability and fine-grain parallelism of synchronous data parallel training. **Presented at MLHPC@SC’19**

Deep Learning Checkpointing with VELOC

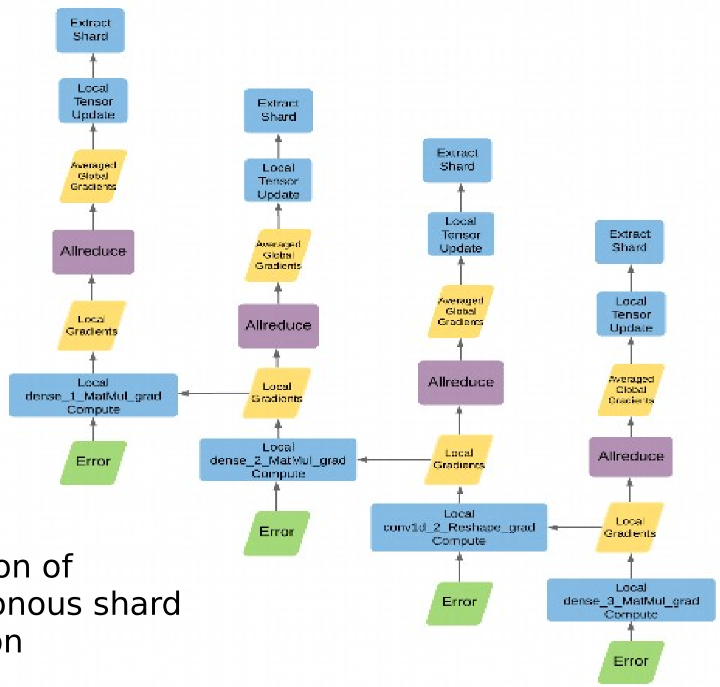
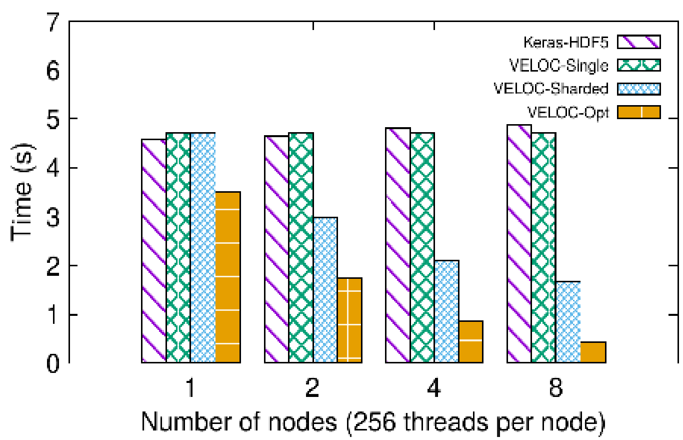


Illustration of asynchronous shard extraction

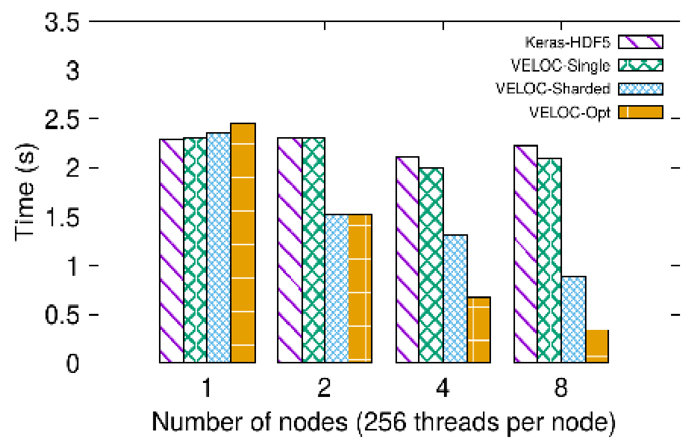
Efficient checkpointing techniques that exploit fine-grain parallelism

- Efficient serialization to local storage
- Sharding: each model replica flushes a different piece of the layer
- Asynchronous shard extraction: serialization of the pieces is embedded in the Tensorflow graph for maximum parallelism

Implementation available for VELOC that integrates seamlessly with Tensorflow 2.0 and Horovod



(b) Blocking phase (lower is better).



(c) Runtime overhead (lower is better).

Bogdan Nicolae, Jiali Li, Justin Wozniak, George Bosilca, Franck Cappello. DeepFreeze: Towards scalable checkpoinitng of deep learning models.
To be presented at CCGrid'20

Conclusions

- VELOC is a checkpoint-restart solution that addresses the problem of high-performance, scalable checkpointing by transparently leveraging heterogeneous storage stacks
- A key strength of VELOC is asynchronous post-processing of checkpoints
  - Techniques to leverage hybrid local storage efficiently for producer-consumer patterns (e.g. flush to PFS)
  - Good support of ECP applications and traditional HPC patterns
  - Experimental support for deep learning
- Use cases beyond resilience:
  - Administrative
  - Productive

- Acknowledgements: Justin Wozniak, Franck Cappello (ANL); JiaLi Li, George Bosilca (UTK)
- Funding: ECP Project, Argonne LDRD

Thank you!

Contact: bogdan.nicolae@acm.org