Update on SoLID Software

Ole Hansen

Jefferson Lab

SoLID Collaboration Meeting January 8, 2021

Prototype Framework Implementation: ARIEL

- "*art* made usable": Easy-to-install bundle of entire *art* suite, independent of custom Fermilab package manager. Installable from source.
- Most recent art version 3.06.03 (Aug 2020) plus dependency packages, integration tests, examples (toyExperiment, art-workbook), container defn's
- Task-based event-level multithreading (TBB)
- Supported on Linux & macOS w/C++17
- Recommended use: Singularity container (see next)
- https://github.com/JeffersonLab/ARIEL

	hannenin Container. Ek turor is be	na vetter vetter	(i) and converter	Art-based integrated event		
0	namenyo consiner: Pa typos in ne	p text 3700007 yesterooy	Opromis	processing energy		
÷	rmike	warkback: Add Studistics' her as reafered location for weekbook on	H MISS 800	Vew loense		
2	container	Container: Pix types in help text	vestorday			
2	marroles	workbook: Fix twos & informational messages	2 months app	Releases		
	externals	Remove packaging wrapper of range-v3 submodule	2 months ago	Q 11aga		
	810	cettils: Handle new CMake policy regarding whitespace in test names	4 days ago	Create a new release		
0	gitiprore	Silence subsence warnings from ONU which if ROOT not set up	2 months ago			
0	gitmodules	Remove packaging wrapper of range-v3 submodule	2 months ago	Languages		
0	LICENSE.md	Update copyright years in LICENSE file	2 months ago	• C++ 03.4% = CMake 0.0%		
0	README.md	Update READMES	last month	Perl 3.1% @ 5.9% @ Shell 3.2% @ Pythen 0.4%		
0	ariel.prg	Add logs. Update READIVE, Add workbook documentation.	3 months ago	Other 0.5%		
0	build-and-install.sh	build scripts: Don't require 'rproc' program to be installed	4 days ago			
0	run-tests.sh	build scripts: Don't require 'riproc' program to be installed	4 days ago			
HE.	omens	S.				

ARIEL Containers

Singularity Container on CUE

```
ifarm1901> module load singularity
ifarm1901> singularity run /group/solid/apps/ARIEL.sif
Singularity> art --version
art 3.06.03
Singularity> ^D
ifarm1901>
```

- Intended as runtime and development environment for SoLID code
- Ubuntu 20.04 LTS base w/gcc 9.3.0. ROOT 6.22.06 built with C++17 support.
- See container help text for instructions:

singularity run-help ARIEL.sif

- Download: https://solid.jlab.org/files/ARIEL.sif (775 MB)
- Docker version planned

Last Meeting's "Next Steps"

- Further evaluate ARIEL: Write test modules & benchmark
- Evaluate JANA2 features
- Develop plan to port simulations to ARIEL with input from GEMC, artg4 and ANL's NPDet/DD4hep
- Investigate deployment on cluster and grid for future simulations (GlueX experience)

Last Meeting's "Next Steps"

• Further evaluate ARIEL: Write test modules & benchmark

• Evaluate JANA2 features

- Develop plan to port simulations to ARIEL with input from GEMC, artg4 and ANL's NPDet/DD4hep
- Investigate deployment on cluster and grid for future simulations (GlueX experience)

A Simple Prototype Module

- Ported benchmarking algorithm from my "parallel Podd" toy analyzer.
- Minimal framework overhead (see screenshot). < 1 hour of beginner-level work.
- This example algorithm implements a CPU-intensive calculation of π [1]

[1] Rabinowitz and Wagon, American Mathematical Monthly, 102 (3), 195-203 (March 1995), doi:10.2307/2975006

```
DetectorTypeC module.cc - emacs@mackinley.redyw.com
    DetectorTypeC: demonstration of a detector with a time-consuming
 // algorithm, simulated here by a calculation O(1000) digits of pi
#include "DetectorTypeCResults.h"
 #include "detail/util.h"
 #include "art/Framework/Core/ReplicatedProducer.h"
 #include "art/Framework/Core/ModuleMacros.h"
 #include "art/Framework/Principal/Event.h"
 #include <iostream>
 #include svectors
 #include <string>
#include <memory>
 #include <cstdlib>
 class DetectorTypeC : public art::ReplicatedProducer {
 nublics
  DetectorTypeC(fhicl::ParameterSet const& pset, art::ProcessingFrame const& frame ):
   void produce( art::Event& event, art::ProcessingFrame const& ) override;
 nrivate
   std::vector<int> m a:
                              // Workspace
                   m result: // Result as string representation of a decimal number
   std::string
   double
                   m scale: // Scale factor
 parallel::DetectorTypeC::DetectorTypeC(fhicl::ParameterSet const& pset,
                                        art::ProcessingFrame const& frame ):
   art::ReplicatedProducer{pset,frame},
   m_scale(pset.get<double>("scale", 1.0))
   produces<DetectorTypeCResults>():
 void parallel::DetectorTypeC::produce( art::Event& event. art::ProcessingFrame const& )
   // This detector type computes n digits of pi
   //..... algorithm goes here .....
   // Create data product
   auto output = std::make unique<DetectorType(Besults>(some result.another result):
   // Add the product to the event
   event.put( std::move(output) );
 DEFINE ART MODULE(parallel::DetectorTypeC)
U:---- DetectorTypeC module.cc All of 1.5k (47.0)
                                                        (C++//] Abbrev)
```

ARIEL Parallel Processing Benchmark

- Run on aonl1 (16 hyperthreaded cores, Intel Xeon E5-2650 v2 @ 2.60GHz), idle conditions
- Admittedly an extreme example: maximally CPU-bound (negligible I/O & memory use)



Comparison: ToyPodd Parallel Processing Benchmark

- Small standalone toy analyzer with hand-implemented multithreading.
- Exact same algorithm & hardware as benchmarked with ARIEL.



Next few months: Continue Prototyping ARIEL

- Custom file format input source module
- DD4hep service (geometry and conditions database), with guidance form ANL's NPDet
- Geant4 producer module (adopted form artg4)
- Rough sketch of data model
- Add art's "studio" SDK & user documentation
- Port of SoLID-GEMC simulations to ARIEL

Estimated SoLID Computing Requirements — preliminary

Run period		2029		2030		2031		2032		2033		2034		Sum
		Spring	Fall	Spring	Fall	Spring	Fall	Spring	Fall	Spring	Fall	Spring	Fall	
Estimated data taking time	calendar days	90	90	70	120	90	90	60	0	90	90	90	68	
Main Experiment		SIDIS-3HeT	SIDIS-3HeT	SIDIS-3HeL	J/Psi	SIDIS-p	SIDIS-p	SIDIS-p	- install -	PVDIS	PVDIS	PVDIS	PVDIS	
Event rate	kHz	100	100	100	60	100	100	100	0	600	600	600	600	
Event size	kB	20	20	20	40	20	20	20	0	2.5	2.5	2.5	2.5	
Uptime		50%	50%	50%	50%	50%	50%	50%	0%	50%	50%	50%	50%	
Raw events	109	389	389	302	311	389	389	259	0	2333	2333	2333	1763	
Raw data rate	MB/s	2000	2000	2000	2400	2000	2000	2000	0	1500	1500	1500	1500	
Raw data to tape	PB	7.8	7.8	6.0	12.4	7.8	7.8	5.2	0.0	5.8	5.8	5.8	4.4	76.7
Simulated events per raw event		10%	10%	10%	10%	10%	10%	10%	10%	5%	5%	5%	5%	
Simulation time per event per core	ms	500	500	500	500	500	500	500	500	500	500	500	500	
Time to analyze one event per core	ms	30	30	30	50	30	30	30	30	10	10	10	10	
Number of passes though data		3	3	3	3	3	3	3	3	3	3	3	3	
Production data reduction factor		5	5	5	5	5	5	5	5	2	2	2	2	
Percentage of data to be processed at JLab		100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	
Simulation CPU requirement	M-core-hours	5.4	5.4	4.2	4.3	5.4	5.4	3.6	0.0	16.2	16.2	16.2	12.2	94.6
Production CPU requirement	M-core-hours	9.7	9.7	7.6	13.0	9.7	9.7	6.5	0.0	19.4	19.4	19.4	14.7	138.9
Total CPU for current experiment (sim+prod)	M-core-hours	15.1	15.1	11.8	17.3	15.1	15.1	10.1	0.0	35.6	35.6	35.6	26.9	233.4
Simulated data to tape	PB	0.8	0.8	0.6	1.2	0.8	0.8	0.5	0.0	0.3	0.3	0.3	0.2	7
Production data to tape	PB	4.7	4.7	3.6	7.5	4.7	4.7	3.1	0.0	8.7	8.7	8.7	6.6	66
Total data to tape (raw+sim+production)	PB	13.2	13.2	10.3	21.2	13.2	13.2	8.8	0.0	14.9	14.9	14.9	11.2	149

Estimated SoLID Computing Requirements (II) — preliminary



- \bullet Raw data rate comparable to GlueX & CLAS12 (~2 GB/s).
- Estimated CPU requirements already manageable with today's farm resources.
- Tape requirements (25–30 PB/yr) significantly higher than current experiments
 → cannot keep all data, should develop data management plan
- J/ Ψ has \sim 50% higher storage requirements due to larger event size (3-fold coincidence)



- Design & development of SoLID software ecosystem in progress
- Actively evaluating prototype implementations
- Anticipated computing resource needs seem mostly manageable at this time