Total presentation time is 7 minutes

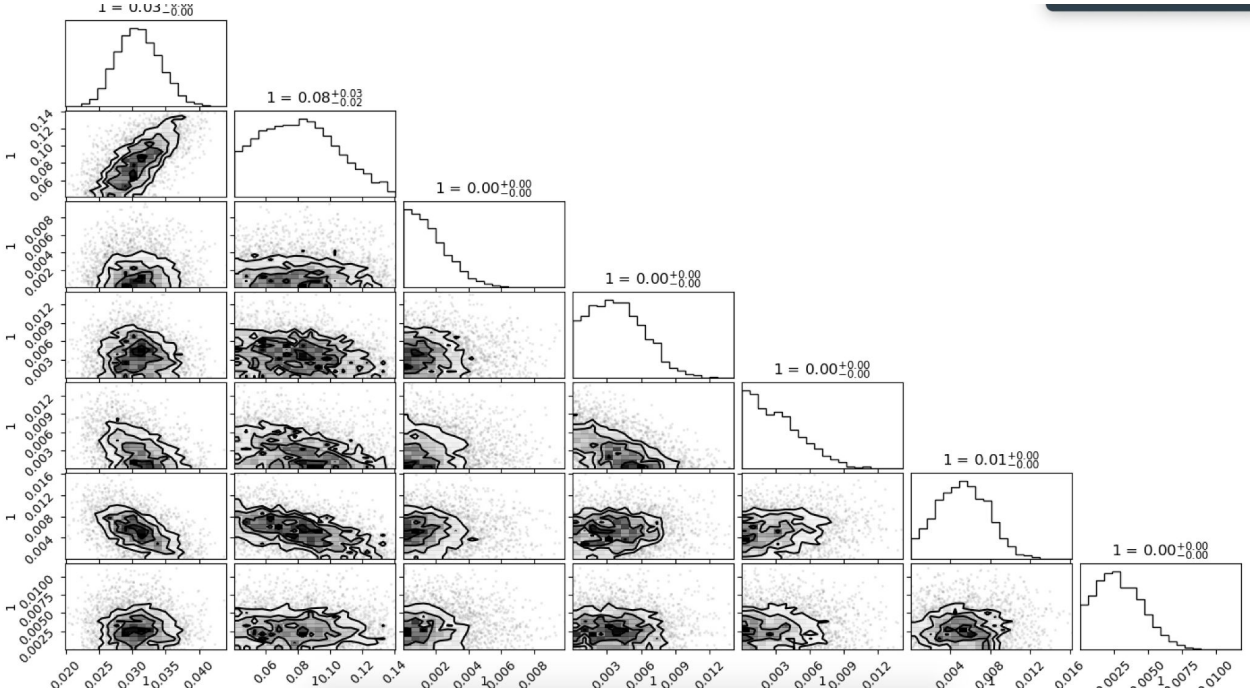
Gaia RVS

Team Members: Ella Wang (ANU)

Mentors: Johan Barthelemy (NVIDIA)
Yuxiang Qin (Uni Melb)

App Name

- All things to do with analysing spectra with errors
  - Pytorch CPU -> GPU
    - Profiling & reading scalability of ngpus
  - Running & parallelizing Fortran code on GPUs
  - Errors: MCMC algorithms & speedup



Evolution and Strategy

- Porting Pytorch code from cpu to gpu
- Finished early, so I also investigated other topics of interest
  - Making Fortran code run faster
  - Sampling posteriors faster

Results and Final Profile

- What were you able to accomplish?
  - Did you achieve speed up?

    CPU: takes hours

    GPU: takes minutes (speedup factor of 10)

    multi GPUs has the potential for even more speedup (depending on utilization), up to factor of 2.5
- What did you learn?
  - Pytorch GPU interfacing
  - Profiling and reading GPU results
  - Different Fortran compilers read different Fortran syntaxes!
  - Nested MCMC sampling is faster and gives Bayesian evidence
  - Enables better testing of hyperparameters
  - Better error estimation!

Energy Efficiency

INPUTS	
# CPU Cores	1
# GPUs (A100)	2
Application Speedup	25.0x

Node Replacement	0.8x

GPU NODE POWER SAVINGS			
	AMD Dual Rome 7742	**8x A100 80GB SXM4**	**Power Savings**
Compute Power (W)	859	6,500	-5,641
Networking Power (W)	36	93	-57
Total Power (W)	**896**	**6,593**	-5,697

Node Power efficiency	**0.1x**

ANNUAL ENERGY SAVINGS PER GPU NODE			
	AMD Dual Rome 7742	**8x A100 80GB SXM4**	**Power Savings**
Compute Power (kWh/year)	7,528	56,940	(49,412)
Networking Power (kWh/year)	318	814	(496)
Total Power (kWh/year)	**7,846**	**57,754**	**(49,908)**

$/kWh	$	0.34
Annual Cost Savings	$	(16,968.60)
3-year Cost Savings	$	(50,905.80)

Metric Tons of CO2	(35)
Gasoline Cars Driven for 1 year	(8)
Seedlings Trees grown for 10 years	(585)

(source: Link)

What problems have you encountered?

- Problems with legacy Fortran code

```
subroutine init_gcr
  use mem_mod
  use input_var_mod,only:input
  implicit none

  character(len=72) :: text
  character(len=9),parameter :: rname='init_gcr'

  ! Check that restart parameter is sensible
  ! res .ge. 1 (number of updates before restarting)
  if (input%gcr_res.lt.1) call stop('gcr_res must be .ge. 1')

  write(text,fmt='(a,i2,a)') &
      'GCR: number of saved vectors = ',input%gcr_res-1,'.'
  call messg(text)
  ! old pops and residual
  call allocate(xold,xs,xe,ys,ye,zs,ze,1,nlevel,rname)
  call allocate(rold,xs,xe,ys,ye,zs,ze,1,nlevel,rname)

  ! conjugate vectors
  if (input%gcr_res .ge. 2) then
      call allocate(pold,xs,xe,ys,ye,zs,ze,1,nlevel,1,input%gcr_res-1,rname)
      call allocate(apold,xs,xe,ys,ye,zs,ze,1,nlevel,1,input%gcr_res-1,rname)
      call allocate(apapold,1,input%gcr_res-1,rname)
      ! Note: lower index -> more recent p, ap
      ! initialize to 0
      pold = 0d0
      apold = 0d0
  endif

  ! initialize ipass = 0
  ipass = 0

  call mem_report('master',rname)
```

```
      IF(IFPRES.EQ.0)GO TO 12
      IF(ITEMP.EQ.1)GO TO 111
C     INTEGRATE EQUATION OF HYDROSTATIC EQUILIBRIUM
      DO 11 J=1,NRHOX
      P(J)=GRAV*RHOX(J)-PRAD(J)-PTURB(J)-PCON
      IF(P(J).GT.0.)GO TO 11
      CALL W(6HJ      ,DFLOAT(J),1)
      CALL W(6HP      ,P,J)
      CALL W(6HPZERO  ,PZERO,1)
      CALL W(6HACCRAD,ACCRAD,NRHOX)
      CALL W(6HPRAD   ,PRAD,NRHOX)
      CALL EXIT
   11 CONTINUE
C  11 P(J)=PTOTAL(J)-PRAD(J)-PTURB(J)
  111 PZERO=PCON+PRADK0+PTURB0
```

Wishlist

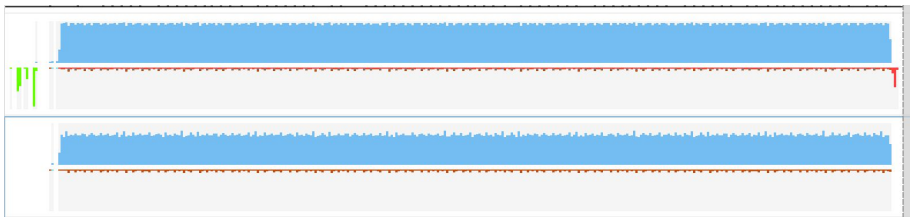
- What do you wish existed to make your life easier?
  - A standard syntax for Fortran between compilers
  - Someone to make this code run faster?

Was it worth it?

- Was this worth it?
  - Yes, it saved a lot of time and gave some ideas I wouldn't have thought of
- Will you continue development?
  - Generating the data (10000 PBS jobs? :))
  - Hyperparameter optimization
- What sustained resources/support will be critical for your work after the event?
  - Gadi time
  - Not critical but some gpu advice from experts would be nice

Application Background

- Charting a 3D map of the Milky Way
- Quicker generation of synthetic spectra used in the analysis
    - NN as emulators
    - Computation on synthetic spectra in Fortran
- Better error analysis
    - Enabled through faster posterior sampling
- Understand the past, present, and future of the Milky Way



Hackathon Objectives and Approach

- Pytorch + GPU
- Nsight & torch.profiler
- Pytorch, cuda, acc, nvidia-hpc-sdk
- scalability tests, parallelization over data vs model

Technical Accomplishments and Impact

- Run Pytorch on GPU + profiling GPU usage
- Using existing methods and tools
- speedup x20
- More hyperparameter tuning & error estimates

Please use 100 words to summarize your team's achievements during this Hackathon

1. Profiled Pytorch model running with GPUs
2. Learned different ways of scaling NNs
3. Faster sampling of posterior for better error estimates

PROMOTING YOUR WORK: AVAILABLE OPPORTUNITIES

- **Papers and Talks:** Please acknowledge the Open Hackathons program and OpenACC Organization in any planned or upcoming papers, presentations, or talks.

  "This work was completed in part at the NCI Open Hackathon, part of the Open Hackathons program. The authors would like to acknowledge OpenACC-Standard.org for their support."
- **Social Media Support:** Please feel free to promote your participation across your social media channels. Tag **@OpenACCorg** and **#OpenHackathons** and we are happy to amplify.
- **Blogs and Technical Write-ups**: Create a blog post or technical article that highlights the work being done and results achieved.
- **Quotes and Testimonials:** Highlight your quote or feedback on our channels (i.e. social, website, etc.).

***Please reach out to Izumi Barker (ibarker@nvidia.com) to discuss marketing options and opportunities.**