Argonne NATIONAL LABORATORY

... for a brighter future

The Asynchronous Dynamic Load-Balancing Library

Rusty Lusk, Steve Pieper, Ralph Butler, Anthony Chan

Mathematics and Computer Science Division

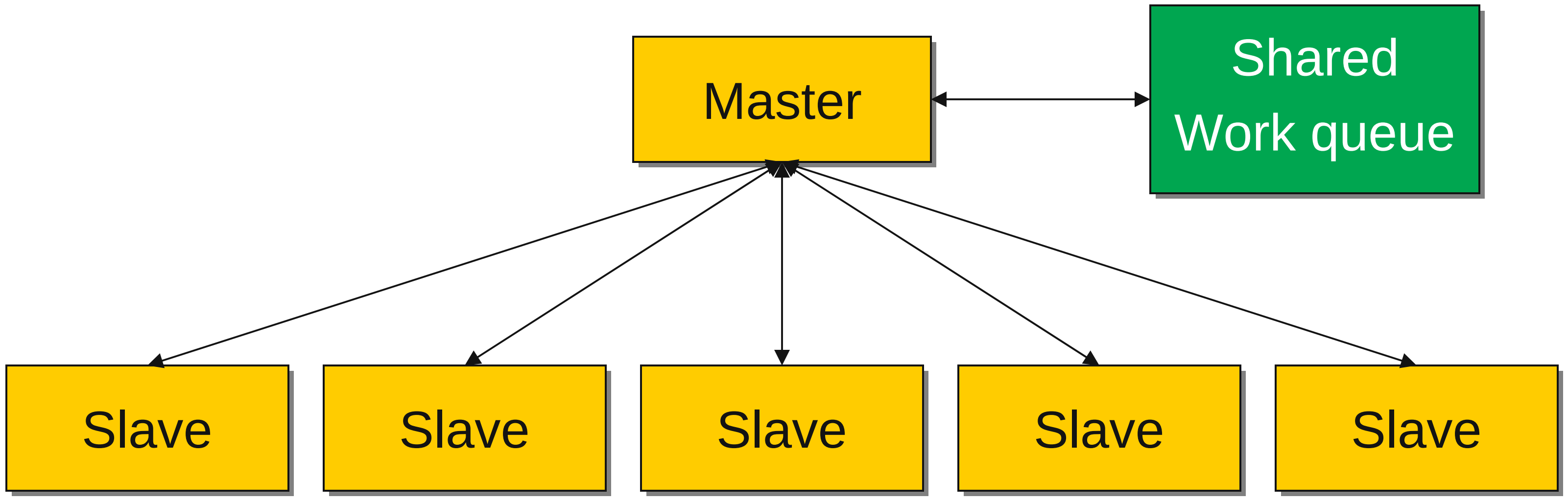
Nuclear Physics Division

Argonne National Laboratory

Outline

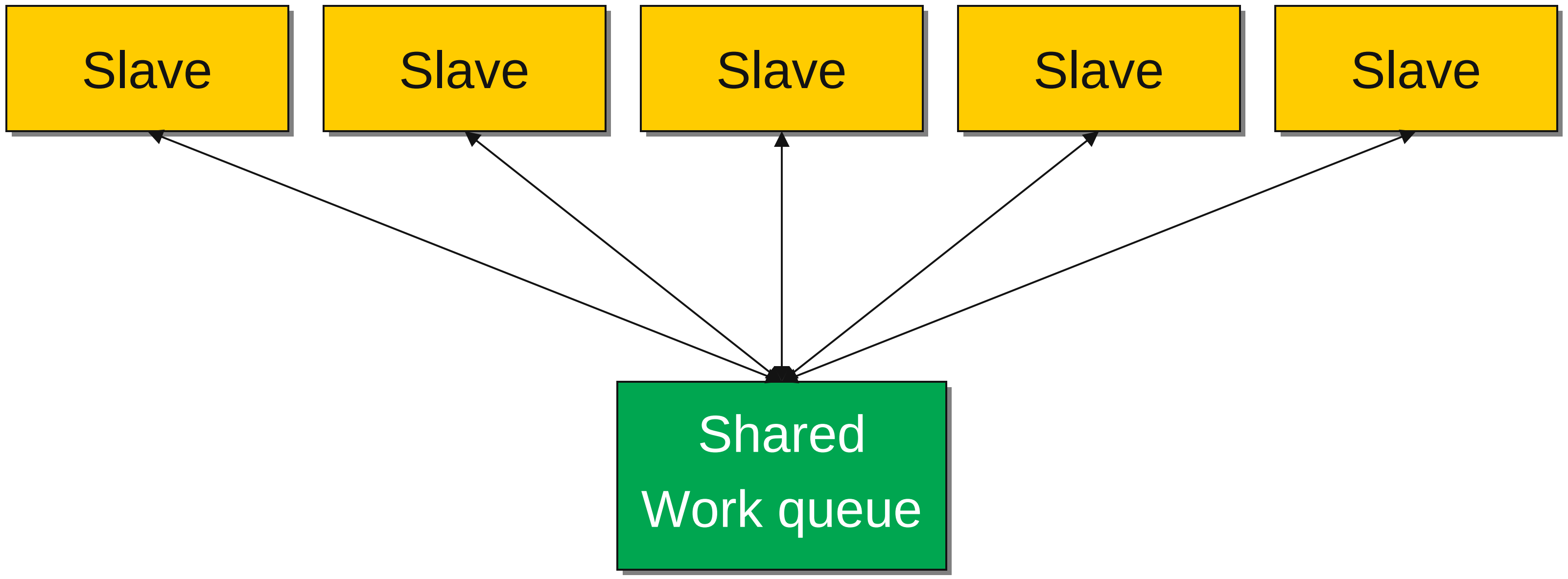
- Overview of ADLB
- The API in a nutshell
- How it works
- Tutorial example
- Activities since last year
- Plans near and far
- The five questions

Master/Slave Algorithms and Load Balancing



- Advantages
  - Automatic load balancing
- Disadvantages
  - Scalability - master can become bottleneck
- Wrinkles
  - Slaves may create new work
  - Multiple work types and priorities that impose ordering

The ADLB Model (no master)



- Doesn’t really change algorithms in slaves
- But need distributed implementation of shared work queue for scalability

The Vision

- No explicit master for load balancing; slaves make calls to ADLB library; those subroutines access local and remote data structures (remote ones via MPI).
- Simple Put/Get interface from application code to distributed work queue hides most MPI calls
  - Advantage: multiple applications may benefit
  - Wrinkle: variable-size work units, in Fortran, introduce some complexity in memory management
- Proactive load balancing in background
  - Advantage: application never delayed by search for work from other slaves
  - Wrinkle: scalable work-stealing algorithms not obvious

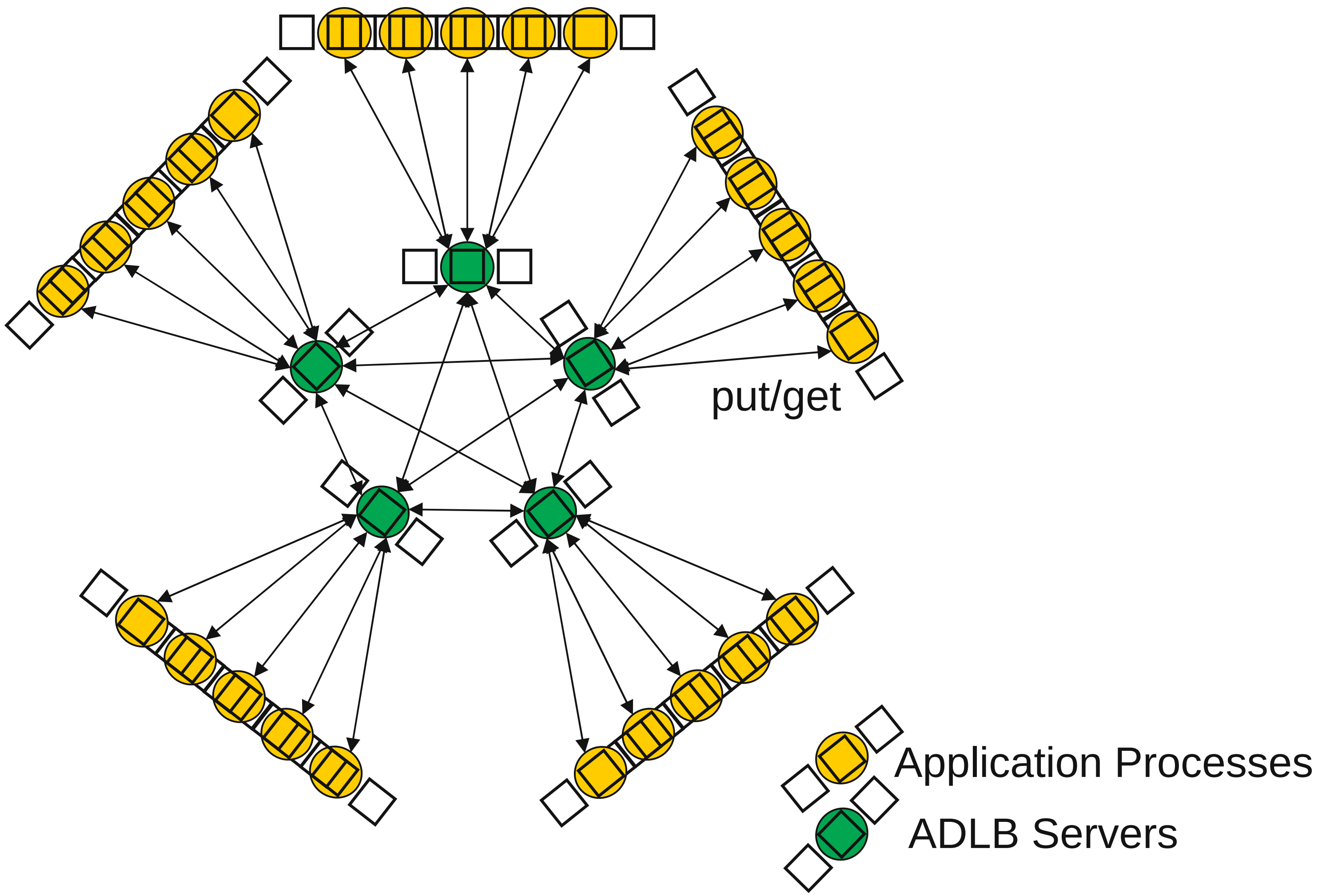
GFMC and ADLB

- Specific Problem: to scale up GFMC, a master/slave code
- General Problem: scaling up the master/slave paradigm in general
  - Usually based on a single (or shared) data structure
- Goal for GFMC: scale to 160,000 processes (available on BG/P)
- General goal: provide simple yet scalable programming model for algorithms parallelized via master/slave structure
- General goal in GFMC setting: eliminate (most) MPI calls from GFMC and scale the general approach
- GFMC is not an easy case:
  - Multiple types of work
  - Any process can create work
  - Large work units (multi-megabyte)
  - Priorities and types used together to specify some sequencing without constraining parallelism (“workflow”)

The API (Application Programming Interface)

- Basic calls
  - ADLB_Init(num_servers, am_server, app_comm)
  - ADLB_Server()
  - ADLB_Put(type, priority, len, buf, answer_dest)
  - ADLB_Reserve(req_types, handle, len, type, prio, answer_dest)
  - ADLB_Ireserve(…)☐
  - ADLB_Get_Reserved(handle, buffer)
  - ADLB_Set_Done()
  - ADLB_Finalize()
- A few others, for tuning and debugging
  - ADLB_{Begin,End}_Batch_Put()
  - Getting statistics

Behind the Scenes



A Tutorial Example: Sudoku

1	2				9			7
		3				6	1	
				7		8		
					5	3		
7		9	1		8	2		6
		5	6					
		1		9				
	6	7				1		
2			5				3	8

Parallel Sudoku Solver with ADLB

1	2				9			7
		3				6	1	
				7		8		
					5	3		
7		9	1		8	2		6
		5	6					
		1		9				
	6	7				1		
2			5				3	8

Work-package =

partially completed “board”

Program:

```
if (rank = 0)
        ADLB_Put initial board

ADLB_Get board
while success  (else done)
    ooh
    find first blank square
    if failure  (problem solved!)
        print solution
        ADLB_Set_Done
    else
        for each valid value
            set blank square to value
            ADLB_Put new board
end while ☐
```

Optimizing Within the ADLB Framework

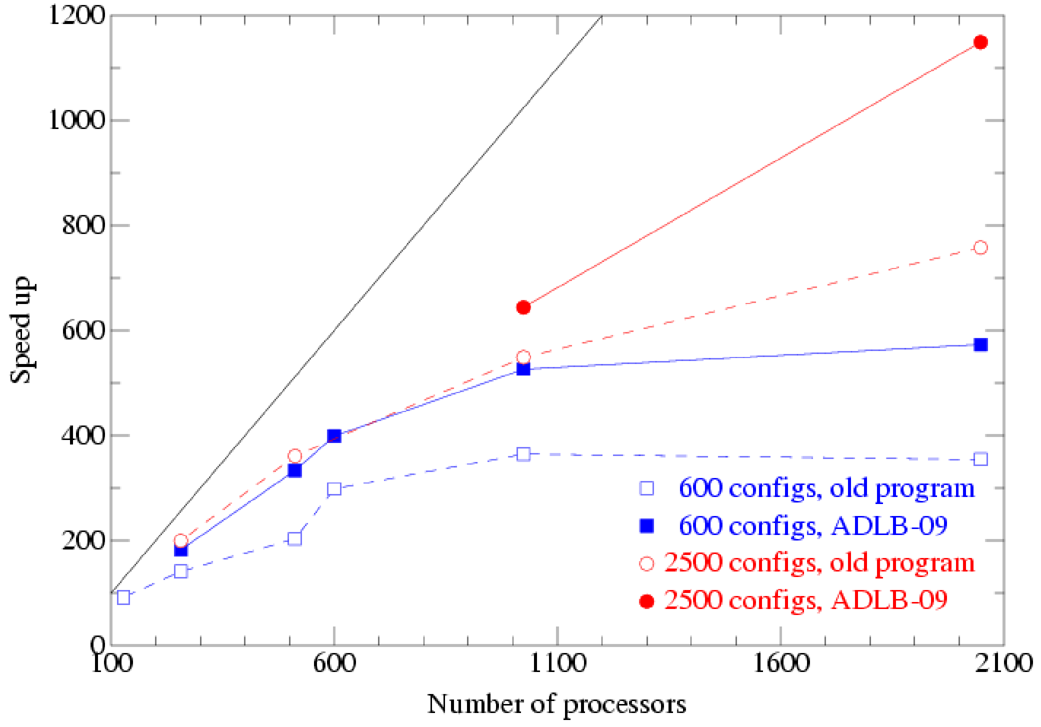
- Can embed smarter strategies in this algorithm (see ooh)
- Even so,☐potentially a *lot* of work packages for ADLB to manage
- Can use priorities to address this problem
  - On ADLB_Put, set priority to the number of filled squares
  - This will guide depth-first search while ensuring that there is enough work to go around
    - *How one would do it sequentially*
- Termination not handled here in case of invalid starting board
  - ADLB must manage internally

Activities Since Last Year

- Ported GFMC and ADLB to BG/P and SiCortex 5832
- Supported runs reported in GFMC talk
  - 14-neutron drop on 16,384 processors of BG/P
  - Speedup of 13,634 (83% efficiency)
  - No microparallization since more configurations
  - ADLB processed 171 million work packages of size 129KB each, total of 20.5 terabytes of data moved
  - Heretofore uncomputed level of accuracy for the computed energy and density
  - Also some benchmarking runs for ^{9}Be and ^{7}Li
- Tuning of ADLB for increased scale required for microparallization
  - Introduction of batches to save memory, reduce status updates
- Load balancing for memory usage
  - Proactive pushing of work units among servers

More Activities

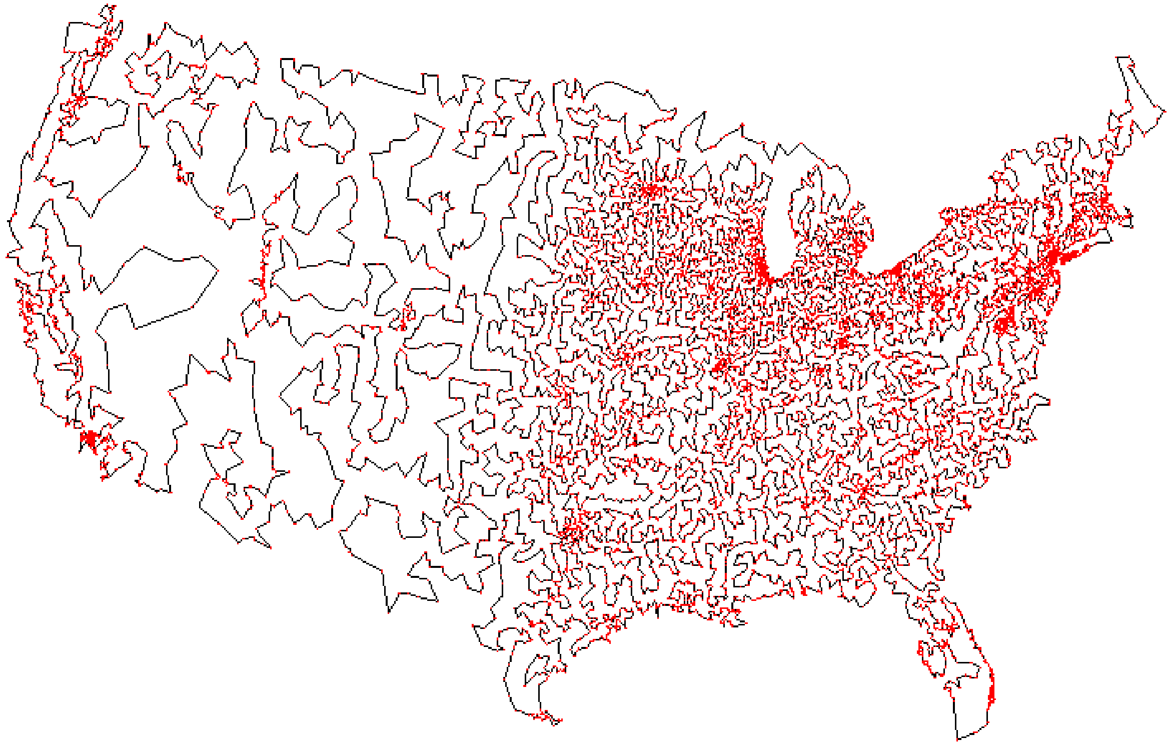
- Debugging support
  - Circular logging buffers
  - The hang detector
- Scaling experiments



- Non-GFMC applications
  - One for testing
  - One real application

A "Classical" CS Test Problem

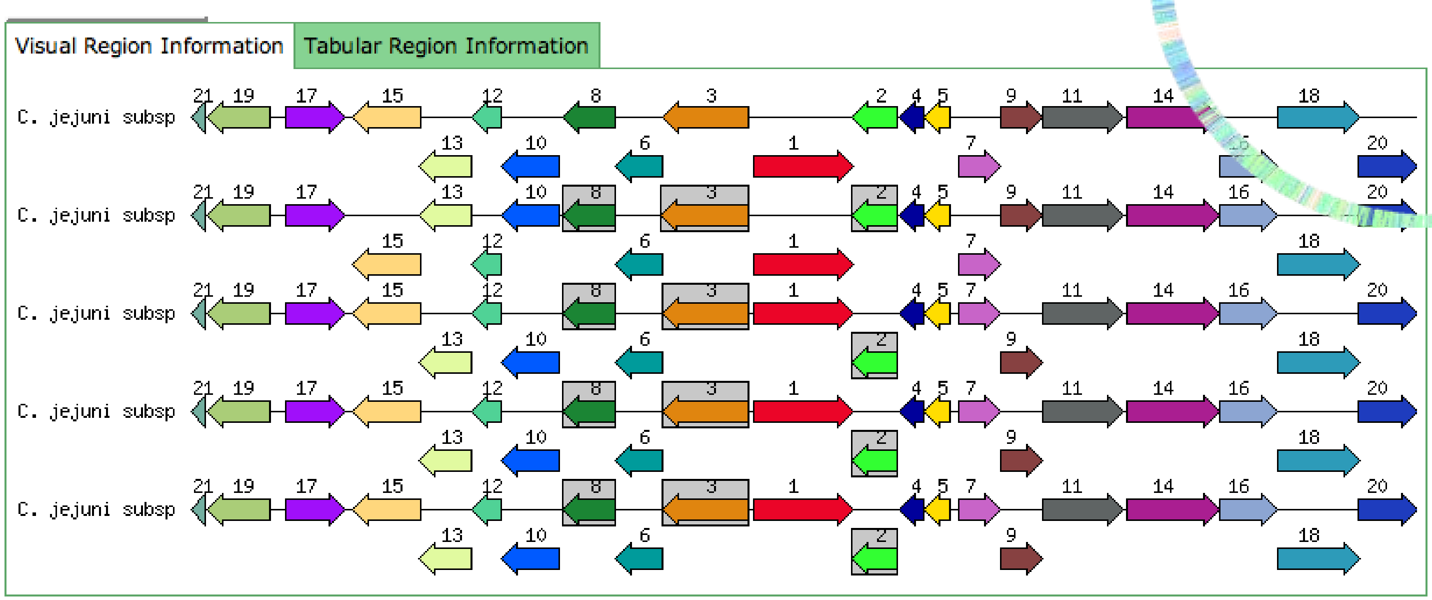
- The "Traveling Salesman" Problem: given a set of points distributed in (two-dimensional) space, find the shortest closed path that goes through each point exactly once.□□□□□□

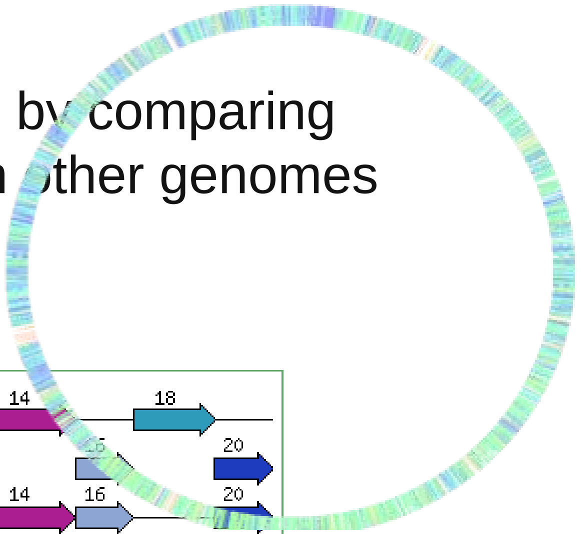


A short path through all cities in the U.S. with population > 500

A Serious Bioinformatics Application

- Identifying genes and their functions in genomes by comparing subsequences with understood subsequences in other genomes



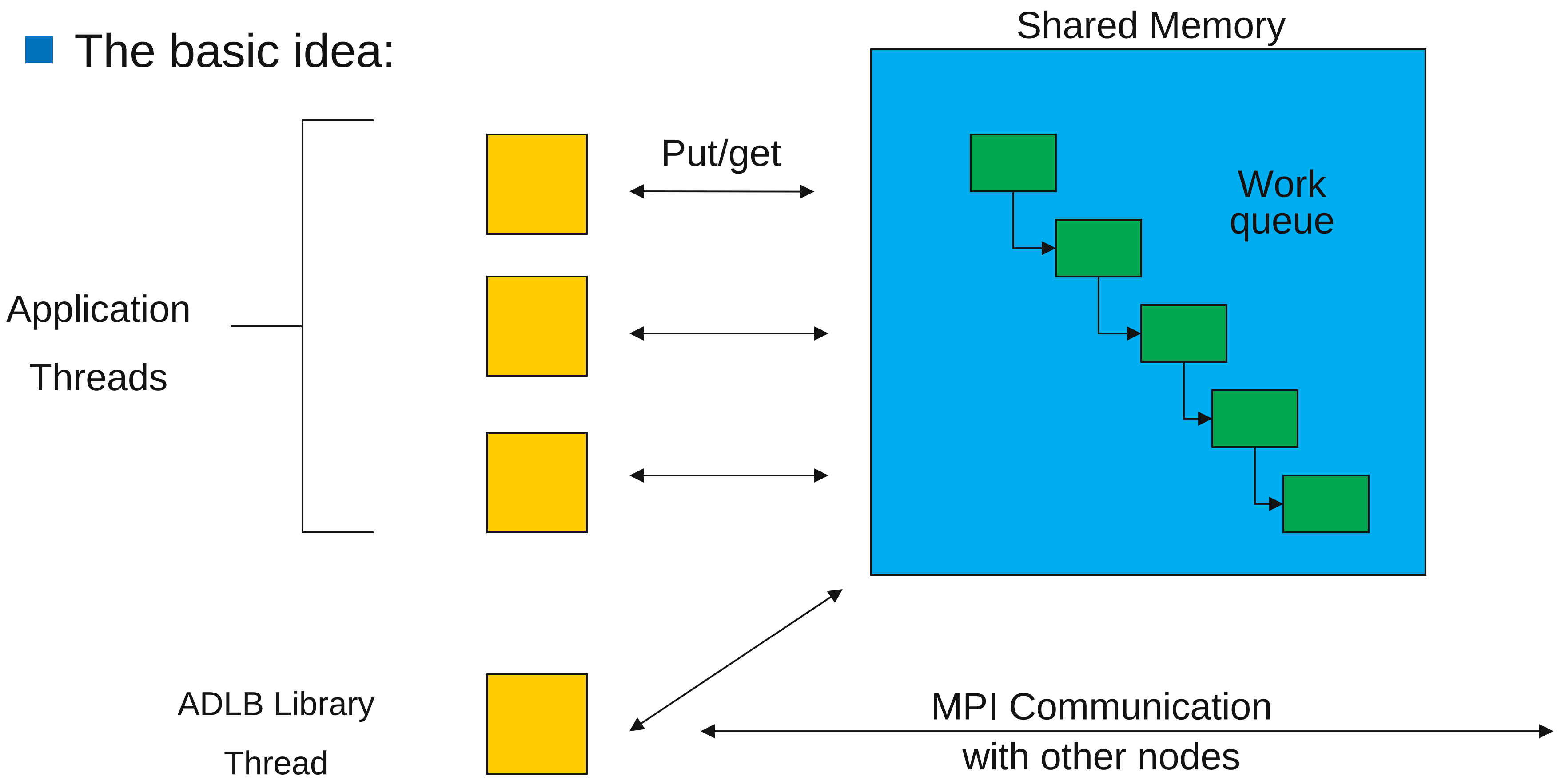


Plans

- Near-term
  - Further microparallelization of GFMC using ADLB, necessary for ${}^{12}C$
  - Scaling (further) up on BG/P
  - Complete the debugging tools
  - Engage other applications, especially within UNEDF
- Long-term
  - Investigate multithreading the application (locally parallel execution of work units via OpenMP)
  - Revisit the thread model for ADLB implementation, particularly in anticipation of BG/Q and experimental compute-node Linux on BG/P
  - Work with others outside the project in other SciDACs

Asynchronous Dynamic Load Balancing - Thread Approach

- The basic idea:

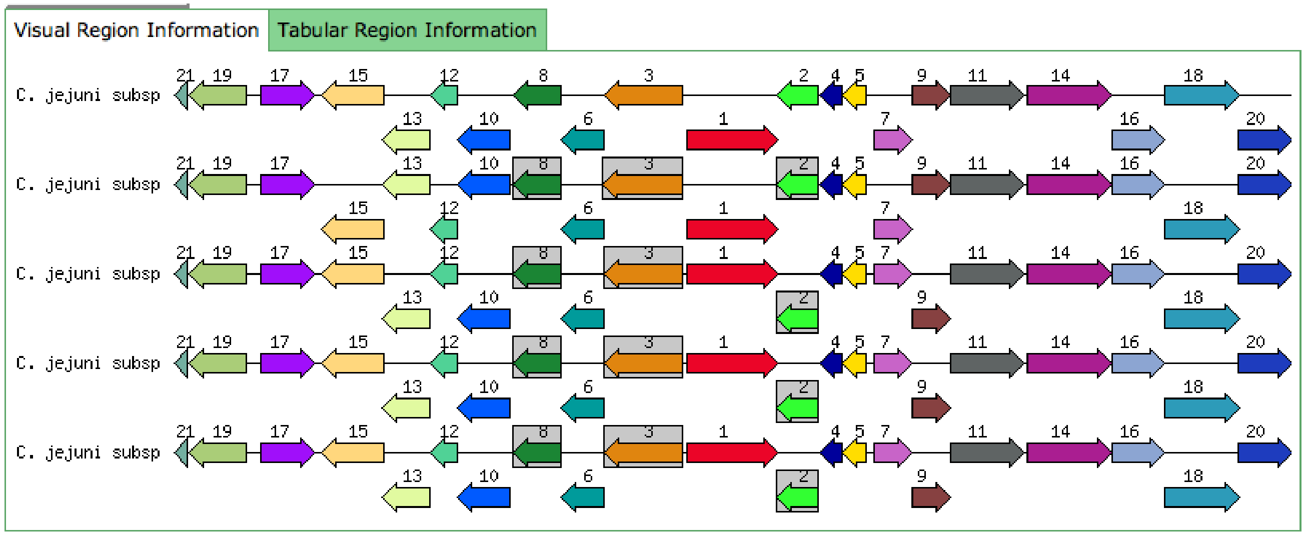


The Questions

- What are the main accomplishments since the last meeting? Is your Year-2 plan well on track?
  - Main accomplishments
    - *complete conversion of GFMC to ADLB (microparallelisation)*
    - *large-scale runs*
  - A little behind on production runs due to debugging problems
- What are the aspects of your science that require high-performance computing?   OR   What problems in high performance computing are you working on in general?
  - Problems in high-performance computing:
    - *How to exploit HPC computers with 100,000 processors*
    - *How to simplify application programming in general*

The Questions (cont.)

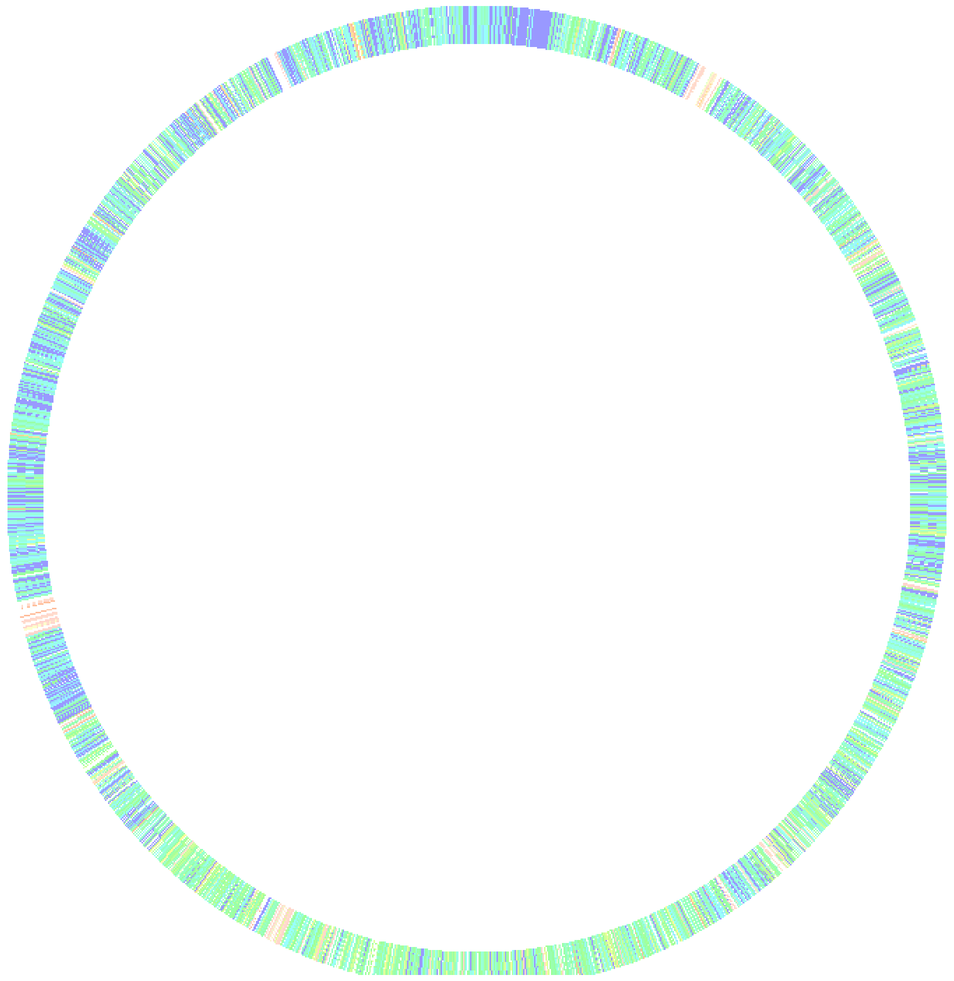
- What are the major computational issues? Are there any questions you would like to bring to the attention of our CS/AM collaborators?   OR   Are there general capabilities of your computer science work that might be of interest to other physicists than the ones you are currently working with?
  - ADLB is a general-purpose library which we are developing / testing / debugging / tuning in the context of GFMC
  - But worth a look for any application in which the parallelism is task-based and there is little communication among the tasks.
- What is the detailed roadmap of your project for the remaining part of Year-2 and Year-3? Could you sketch the work plan for Years 4 and 5?
  - Near-term: scaling up and doing ^{12}C
  - Far-term: threads to allow further scaling, both in app and lib
- Are there any "showcase" (i.e., of Nature/Science caliber)physics and computational questions that you are hoping to answer in Years 2 and 3?
  - It's up to Steve!

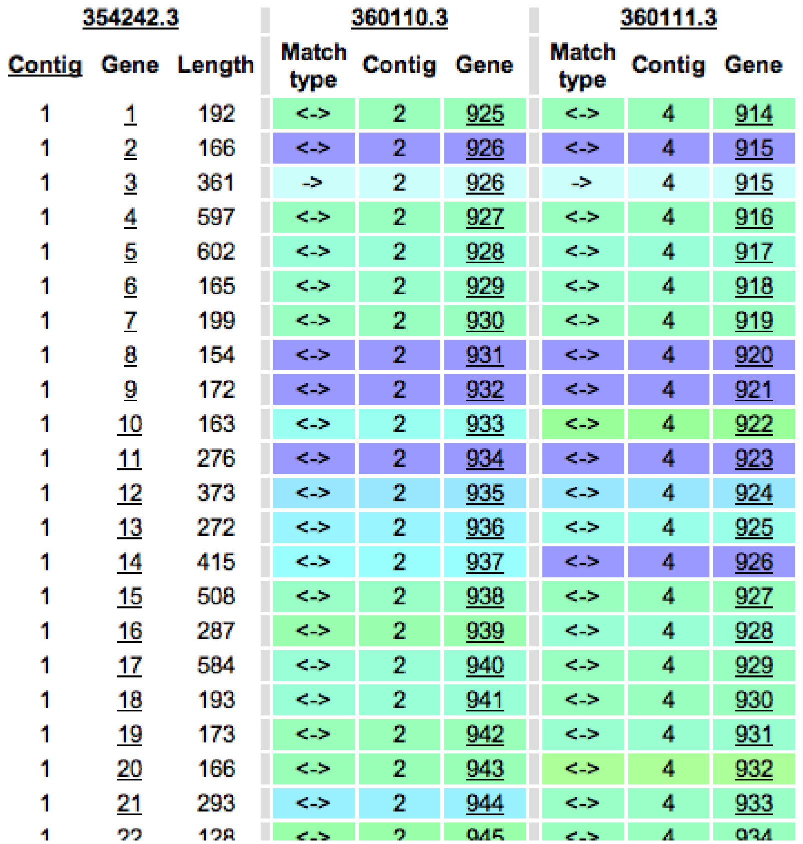


Clustering of functionally related genes. This diagram shows a graphical display of similar chromosomal regions of five genomes with the highest score, based on similar proteins in this region, and phylogenetic distance. The display is centered on this focus PEG, which is shown in red and numbered 1. In this example, highly similar genes from various *Campylobacter jejuni* strains are 'pinned through a biotin biosynthesis gene. Surrounding genes are also involved in this biotin biosynthetic pathway.

Campylobacter jejuni is a bacterial pathogen that is one of the most common causes of human gastroenteritis in the world.

Biotin is vitamin B7 which is necessary for the production of fatty acids (for cell walls/membranes) and he metabolism of fats and amino acids.





354242.3			360110.3			360111.3		
Contig	Gene	Length	Match type	Contig	Gene	Match type	Contig	Gene
1	1	192	<->	2	925	<->	4	914
1	2	166	<->	2	926	<->	4	915
1	3	361	->	2	926	->	4	915
1	4	597	<->	2	927	<->	4	916
1	5	602	<->	2	928	<->	4	917
1	6	165	<->	2	929	<->	4	918
1	7	199	<->	2	930	<->	4	919
1	8	154	<->	2	931	<->	4	920
1	9	172	<->	2	932	<->	4	921
1	10	163	<->	2	933	<->	4	922
1	11	276	<->	2	934	<->	4	923
1	12	373	<->	2	935	<->	4	924
1	13	272	<->	2	936	<->	4	925
1	14	415	<->	2	937	<->	4	926
1	15	508	<->	2	938	<->	4	927
1	16	287	<->	2	939	<->	4	928
1	17	584	<->	2	940	<->	4	929
1	18	193	<->	2	941	<->	4	930
1	19	173	<->	2	942	<->	4	931
1	20	166	<->	2	943	<->	4	932
1	21	293	<->	2	944	<->	4	933
1	22	128	<->	2	945	<->	4	934

Sequence similarity comparison of genomes. These genomes are fairly similar as noted by the purple, blue and green gene colors. The comparison is in reference to the outside genome (circle). Comparison of all proteins in 2 *Campylobacter jejuni* genomes. The results are presented in a table, with different genomes shown side-by-side in columns, and proteins in rows. The proteins are listed in order of their appearance in the selected reference genome.

Argonne National Laboratory