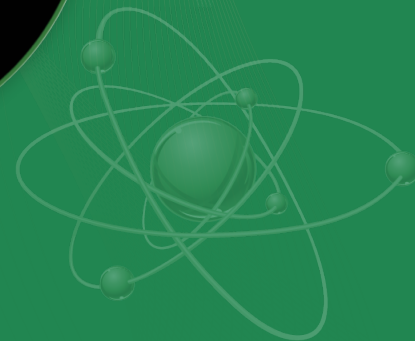
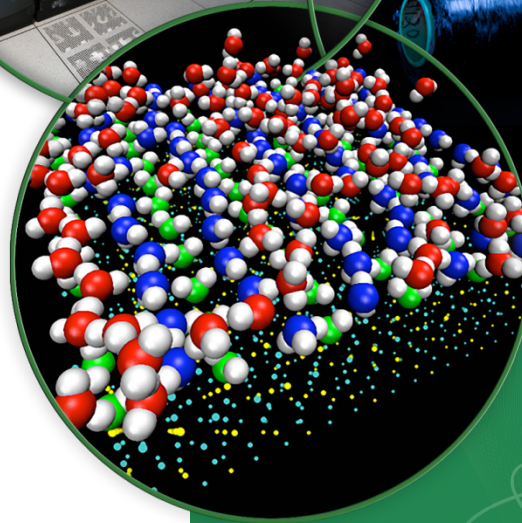
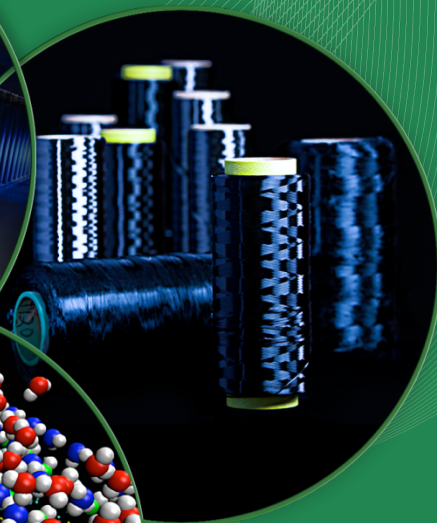


Graph500 in OpenSHMEM

Ed D'Azevedo

Neena Imam

Oak Ridge National
Laboratory



Acknowledgements



This work was supported by the United States Department of Defense (DoD) and used resources of the DoD-HPC Program at the Oak Ridge National Laboratory and at the Oak Ridge Leadership Computing Facility.

Background

- One-sided programming approach is used in unstructured graph algorithms for large-scale data processing
- MPI-2 offers MPI_Put()/MPI_Get() to data in “windows” to memory
- How does MPI-2 compare with OpenSHMEM in implementing a graph algorithm such as Breadth-First Search (BFS) on a large graph?
- Implement Graph500 using **same** code base for MPI-2 one-sided implementation of Graph500

Graph 500 Benchmark

- Benchmark representative of data intensive supercomputer applications. Conceptually similar to numerical intensive High Performance Linpack (HPL) for ranking Top500 computers
- Stresses communication subsystem and non floating-point operations
- Implementations in octave, Cray-XMT, C+OpenMP, C+MPI1, C+MPI-2 (one-sided)

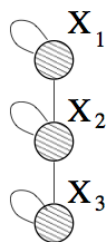
Graph generation

- Scalable Kronecker graph generator for undirected graph
 - Vertices = 2^{scale}
 - Edges = Vertices * edgefactor, edgefactor = 16 by default
 - scale=26 (17.2 GBytes), scale=29 (137.4 GBytes), scale=32 (1099.5 GBytes)
- Generate edges in parallel (may have duplicates and self loops)
- Construct graph (compressed sparse row storage) from edge list
- Need alltoallv operation in SHMEM

Kronecker product of matrices

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \doteq \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \dots & a_{n,m}\mathbf{B} \end{pmatrix}$$

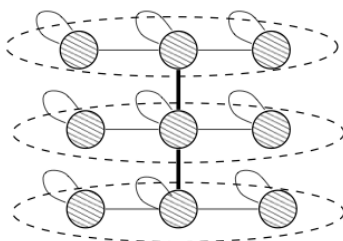
Kronecker product of graphs



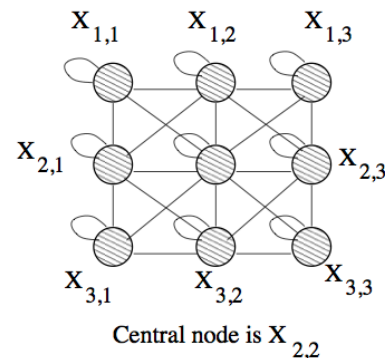
(a) Graph K_1

1	1	0
1	1	1
0	1	1

(d) Adjacency matrix
of K_1



(b) Intermediate stage

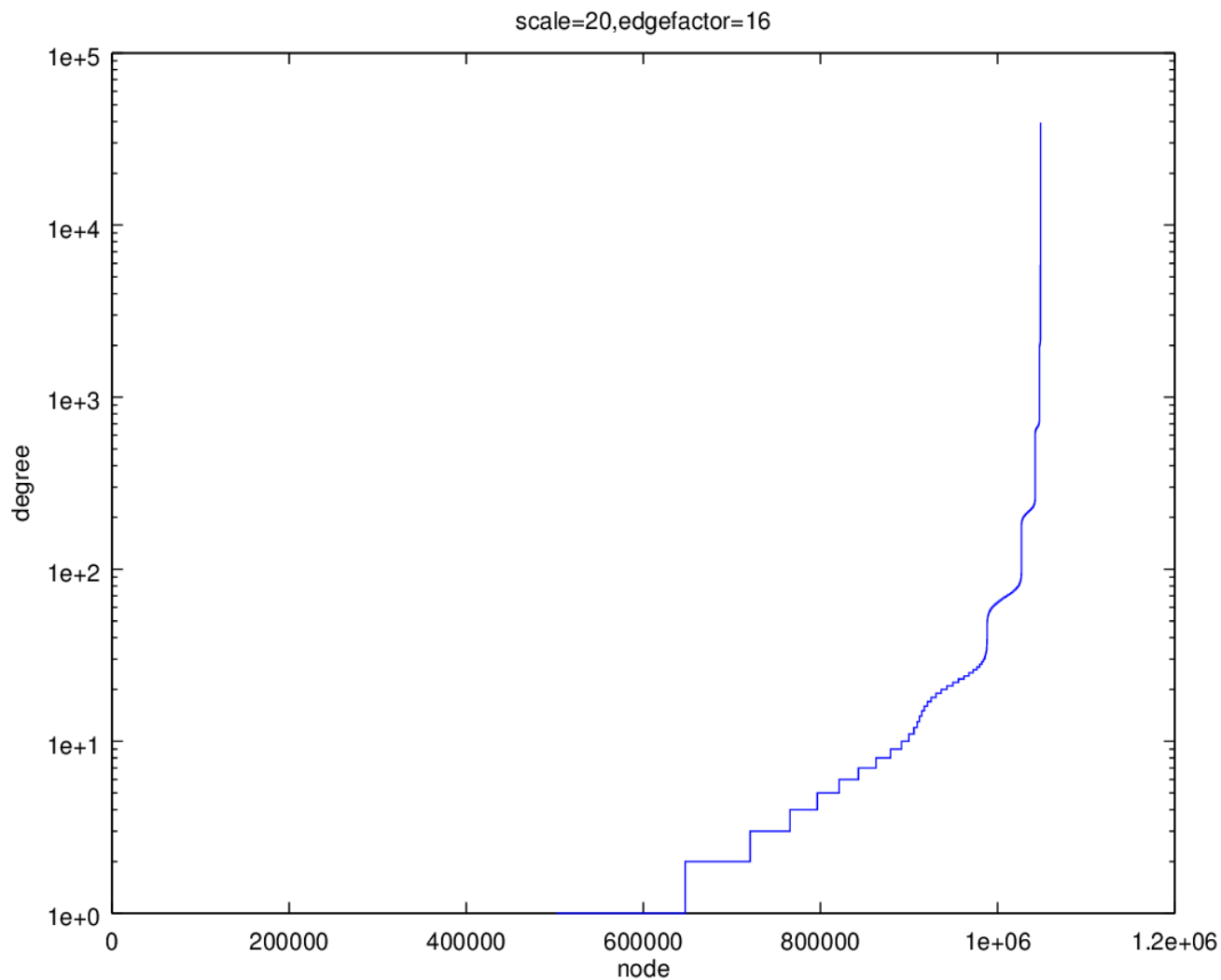


(c) Graph $K_2 = K_1 \otimes K_1$

K_1	K_1	0
K_1	K_1	K_1
0	K_1	K_1

(e) Adjacency matrix
of $K_2 = K_1 \otimes K_1$

Degree distribution, scale=20



BFS starting with random root

- Start with single root, mark neighbor vertices (level 1), then neighbors of neighbors (level 2), ..., until all vertices in connected component are visited
- Outcome is list of predecessor or parent vertices
- Validation phase to check correctness:
 - no cycles (form a tree)
 - predecessor vertex differ by 1 level
 - all vertices in connected component are visited
 - vertex and parent are connected by an edge in graph

Details in one-sided implementation

- Two distributed bit arrays (one new, one old), implemented as unsigned long, to mark vertices that are processed. Each bit may represent 4 vertices
- Two distributed predecessor arrays (unsigned long), upper 16 bits are used to encode level
- MPI Accumulate operations:
 - MPI_MIN on MPI_INT64_T
 - MPI_BOR (bit-wise OR) on MPI_UNSIGNED_LONG
- Many outstanding MPI_Get and MPI_Put operations
- Need SHMEM atomic operations (MIN, bit-wise OR)
- Need SHMEM operations on *unsigned* long data type

Details on validation

- Processing in chunks (chunksize = 2^{22})
- MPI_Allreduce operations:
 - MPI_LOR (logical or), MPI_AND (logical and) on MPI_INT, emulated using global bitwise operation shmem_int_or_to_all() and shmem_int_and_to_all()
 - MPI_MAX on MPI_UINT64_T
- Need SHMEM MAX reduction on unsigned long

Details on SHMEM implementation

- MPI_Put()/MPI_Get() implemented using shmем_putmem()/shmем_getmem()
- MPI_Win_fence() implemented using shmем_barrier_all()
- Operations on unsigned long integer emulated by long integer
- Atomic operations such as MIN, bit-wise OR emulated by atomic conditional swap (CSWAP) on long integer
- Note shmем_long_fadd() (atomic add by zero) to get current value

Routine shmem_long_min

```
1 void shmem_long_min (long *gvar, long value, int pe)
2 {
3     long cval = 0;
4     long lval = 0;
5     int is_done = 0;
6
7     assert( (0 <= pe) && (pe < shmem_n_pes()) );
8
9     lval = shmem_long_fadd( gvar, (long) 0, pe );
10    if (value < lval) {
11        do
12        {
13            cval = shmem_long_cswap (gvar, lval, MIN (lval, value), pe);
14            is_done = (cval == lval) || (cval <= value);
15            lval = cval;
16        }
17        while (!is_done);
18    };
19 }
```

Routine shmем_ulong_bor

```
void shmем_ulong_bor(unsigned long *gvar, unsigned long value, int pe)
{
    /*
     * perform Bitwise OR
     */
    long cval = 0; long lval = 0; unsigned long ulval = 0;
    unsigned long new_ulval = 0; long new_lval = 0; int is_done = 0;

    lval = shmем_long_fadd( (long *) gvar, (long) 0, pe );
    do
    {
        memcpy( &ulval, &lval, sizeof(ulval) );
        new_ulval = ulval | value;
        if (new_ulval == ulval) {
            /* bits set already */
            break;
        };
        memcpy( &new_lval, &new_ulval, sizeof(new_lval) );

        cval = shmем_long_cswap ((long *) gvar, lval, new_lval, pe);
        is_done = (cval == lval);
        lval = cval;
    }
    while (!is_done);
}
```

Experiments

- Bazooka SGI cluster, 12-node SGI Altix XE1300 cluster. Each node has two 2.8 Ghz/12M 6-core Xeon X5660 (total 12 physical cores, **24** hyper-thread cores), **48** GBytes of 1333MHz DDR, Mellanox ConnectX QDR Infiniband. SGI MPT shmem 2.03
- Cray XK7 Chester cluster in Oak Ridge Leadership Computing Facility (OLCF), 96 compute nodes, each node has 32 Gbytes RAM, one **16**-core AMD Opteron 6200 Interlagos processor and Nvidia K20X GPU, Cray Gemini network, Cray SHMEM and Cray MPI

SGL cluster

	MPI2	Scale=20	1 node	SHMEM		
task	generation	construction	BFS median	generation	construction	BFS median
8	6.3	1.5	2.7	6.3	1.7	1.3
12	6.3	9.7	2.3	6.3	1.9	1.2
16	7.0	2.0	1.8	7.9	2.0	0.9
24	7.0	1.7	1.3	7.9	1.9	0.7

- 8 nodes, scale=26, 192 tasks (24 tasks per node)
 - generation time: MPI2 13.1, SHMEM 13.1
 - Construction time: MPI2 5.5, SHMEM 9.8
 - BFS median: MPI2 N/A, SHMEM 808.4

MPI-2 One-sided on 24 tasks scale=26

- MPI_Put()/MPI_Get() may be buffered until MPI_Win_fence()
- MPI-2 can fail on large problems due to out of memory error or too many outstanding requests

```
mpirun bazooka02,bazooka03,bazooka10,bazooka11,bazooka06,bazooka07,bazooka08,bazooka09 24 ../graph500_mpi_one_sided 26 16
graph_generation:      13.095883 s
construction_time:     5.594882 s
Running BFS 0
*** MPI WARNING: MPI has run out of request entries.
*** The maximum allowed value is: 1048576
*** The current allocation level is:
***     MPI_REQUEST_MAX = 16384
*** MPI WARNING: MPI has run out of request entries.
*** The maximum allowed value is: 1048576
*** The current allocation level is:
***     MPI_REQUEST_MAX = 16384
MPI: Global rank 148 is aborting with error code 0.
      Process ID: 22144, Host: bazooka08, Program: /autofs/nccs-svm1_home1/efdazedo/WORK/Graph/Graph500/graph500_openshmem/mpi/graph500_mpi_one_sided
```

Cray XK7, 16 nodes

	MPI2		Scale=20	SHMEM		
tasks	generation	construction	BFS median	generation	construction	BFS median
64	14.4	2.0	11.1	14.9	1.8	3.4
128	15.1	2.2	6.7	14.9	2.0	2.2
256	19.1	3.0	3.0	20.5	2.5	1.8

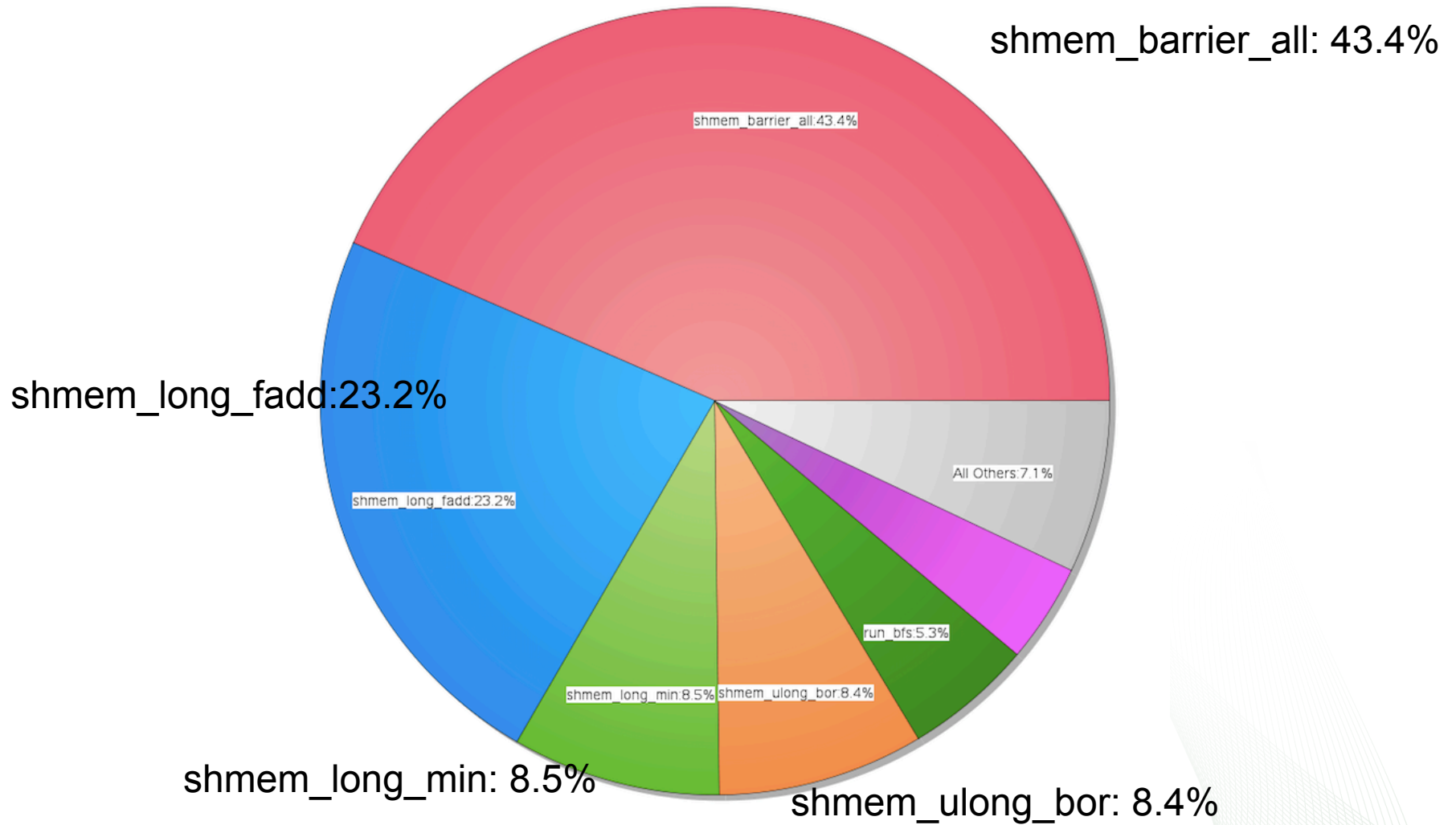
	SHMEM	Scale=26	
tasks	generation	construction	BFS median
64	38.3	14.9	187.8
128	19.8	8.7	97.9
256	26.4	8.1	57.7

Cray XK7, 64 nodes

SHMEM	64 nodes	Scale=28	
tasks	generation	construction	BFS median
128	83.8	30.7	379.9
256	42.3	17.4	208.4
512	22.0	10.4	120.6
1024	31.6	10.2	79.4

- 64 nodes, 512 tasks, scale=30
 - Graph generation 93.9
 - Construction 41.9
 - BFS median 426.4

Exclusive Samples



Wallclock time: 228.641307s

Call tree

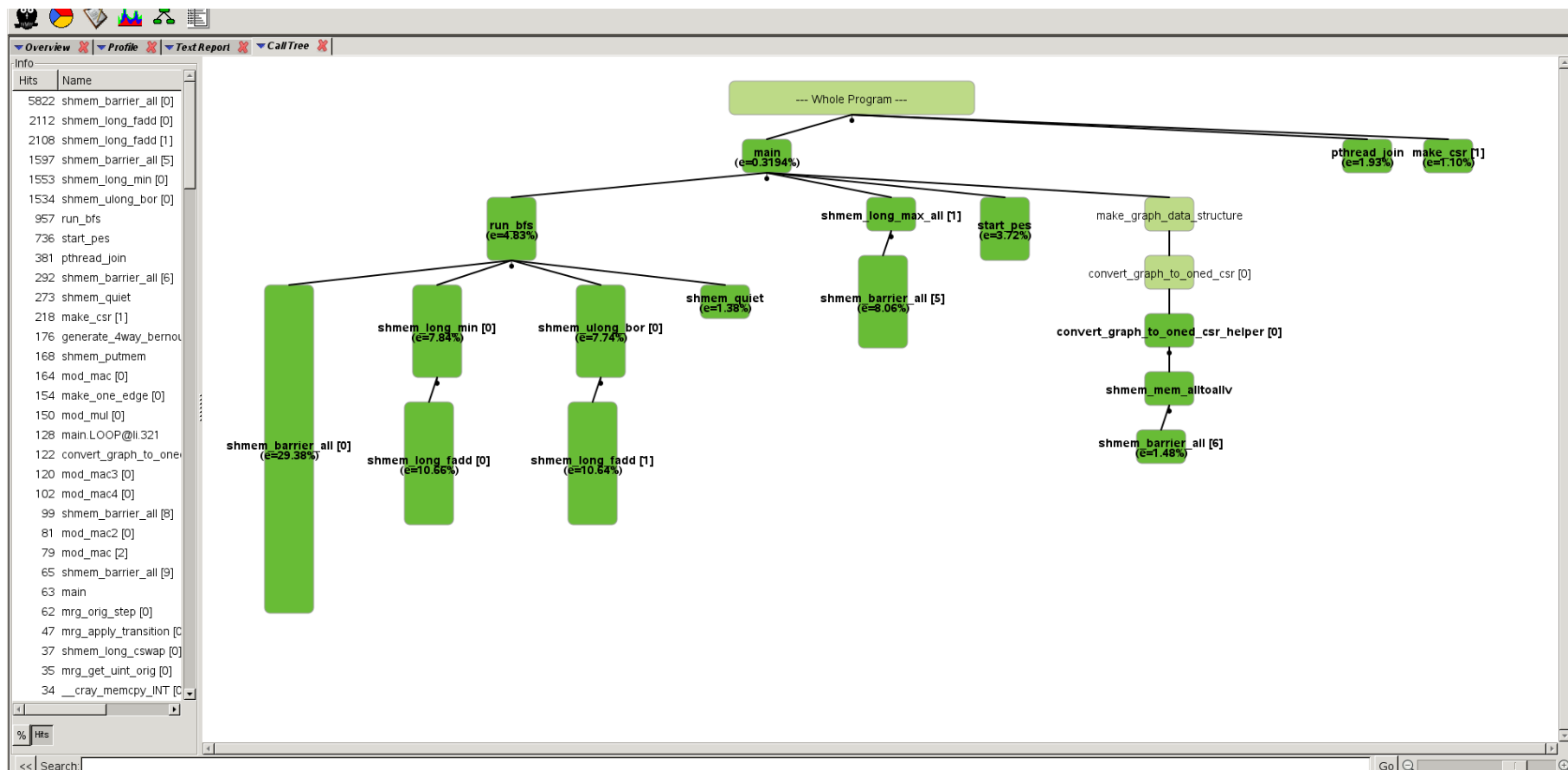


Table from craypat

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function PE=HIDE Thread=HIDE
100.0%	19856.5	--	--	Total
83.3%	16542.6	--	--	SHMEM
39.8%	7899.0	2669.0	25.3%	shmem_barrier_all
21.3%	4221.3	1773.7	29.6%	shmem_long_fadd
7.8%	1553.8	739.2	32.3%	shmem_long_min
7.7%	1535.0	778.0	33.7%	shmem_ulong_bor
3.7%	736.8	623.2	45.9%	start_pes
1.4%	273.4	181.6	40.0%	shmem_quiet
14.4%	2862.9	--	--	USER
4.8%	957.1	358.9	27.3%	run_bfs
1.3%	252.2	310.8	55.3%	mod_mac
1.1%	219.1	65.9	23.1%	make_csr
1.0%	198.0	265.0	57.3%	mod_mul
1.9%	381.8	472.2	55.4%	PTHREAD
1.9%	381.8	472.2	55.4%	pthread_join

Summary

- Graph 500 BFS one-sided implementation using OpenSHMEM 1.0
- Need memory efficient alltoallv operation
- Atomic operations such as MIN, bit-wise OR
- Operations on unsigned long integer type
- Current implementation of BFS one-sided in MPI-2 has issue with too many out standing requests
- SHMEM implementation competitive with MPI-2 and scalable to 1024 tasks

Questions?

