



TSHMEM: Shared-Memory Parallel Computing on Tiler Many-Core Processors



IPDPS/HIPS'13

UF UNIVERSITY of
FLORIDA



THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Virginia
Tech
VIRGINIA POLYTECHNIC INSTITUTE
AND STATE UNIVERSITY

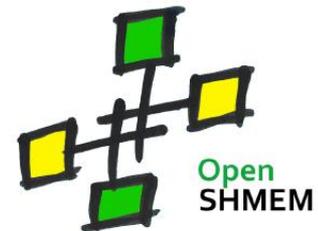
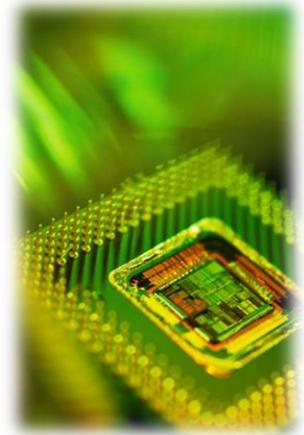
BYU
BRIGHAM YOUNG
UNIVERSITY

Bryant C. Lam (speaker)
Alan D. George
Herman Lam

*NSF Center for High-Performance Reconfigurable
Computing (CHREC), University of Florida*

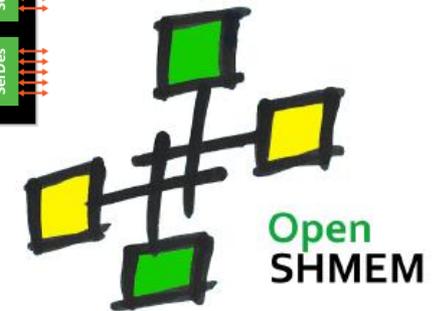
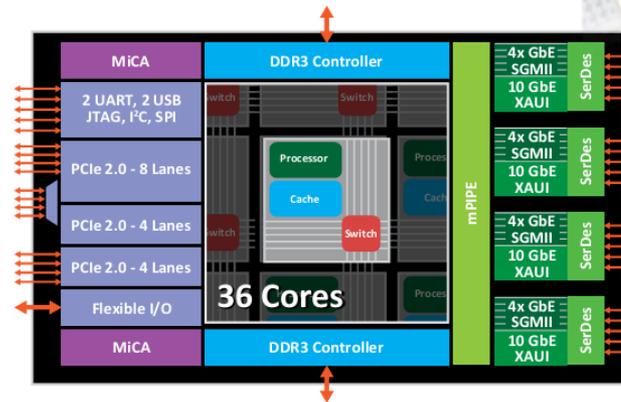
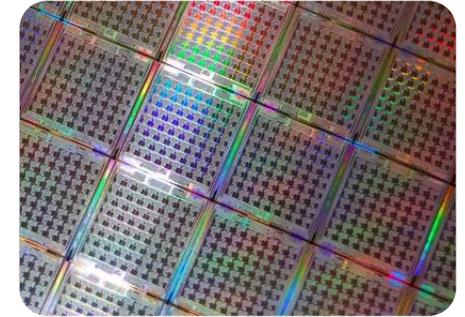
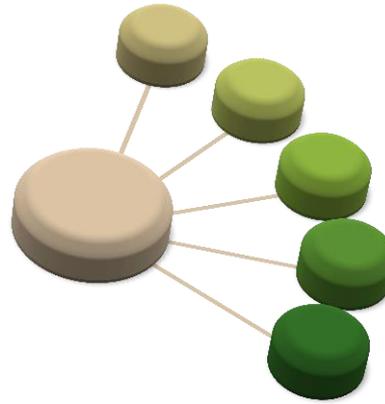
Motivation

- Emergent many-core devices vary considerably in architecture and performance characteristics
 - e.g., scalable mesh networks, ring-bus topologies
- Platforms provide separate APIs to individually leverage underlying hardware resources
 - Difficult to develop efficient cross-platform apps
- **Objective**
 - Bridge the gap between many-core hardware utilization and developer productivity
- **Approach**
 - **Quantify performance** of many-core devices via application kernels and microbenchmarks
 - **Develop TSHMEM** to increase device utilization of various many-core platforms with arch-specific insights



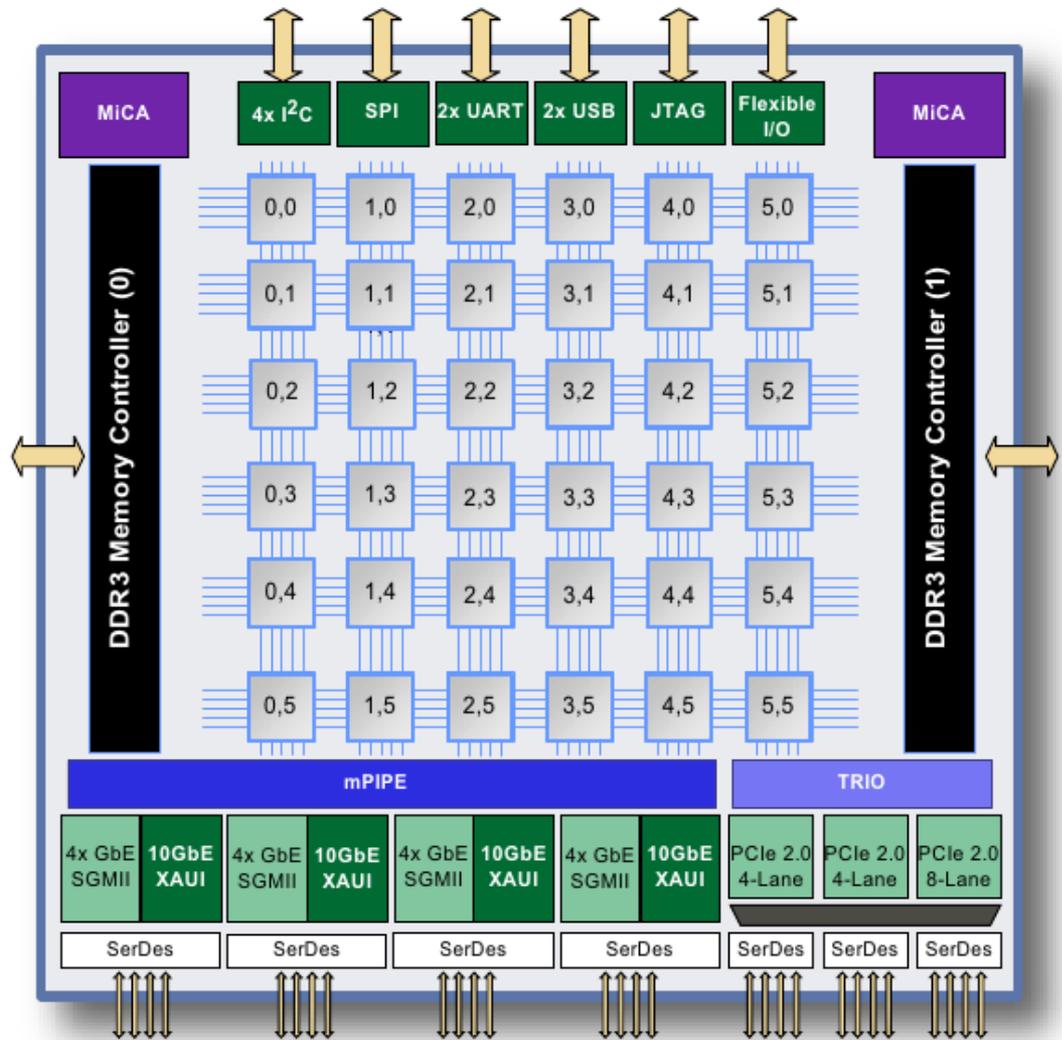
Outline

- Tileria Many-Core
 - Chip Architectures
 - Microbenchmarks
 - Shared Memory Copy
 - UDN Latency
 - TMC Spin/Sync Barriers
- TSHMEM Computing
 - Design Infrastructure
 - Performance Analysis
 - Put/Get Transfers
 - Barrier Synchronization
 - Collectives
 - Application Case Studies
- Conclusions, Q&A



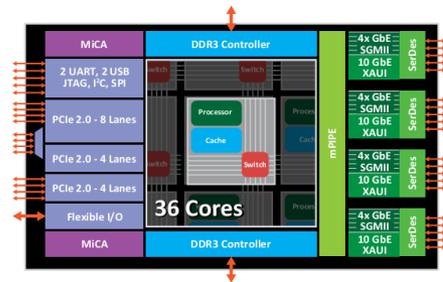
Tilera TILE-Gx Architecture

- 64-bit VLIW processors
- 32k L1i cache, 32k L1d cache
- 256k L2 cache per tile
- Up to 750 BOPS
- Up to 200 Tbps of on-chip mesh interconnect
- Over 500 Gbps memory bandwidth
- 1 to 1.5 GHz operating frequency
- Power consumption: 10 to 55W for typical applications
- 2-4 DDR3 memory controllers
- mPIPE delivers wire-speed packet classification, processing, distribution
- MiCA for cryptographic acceleration



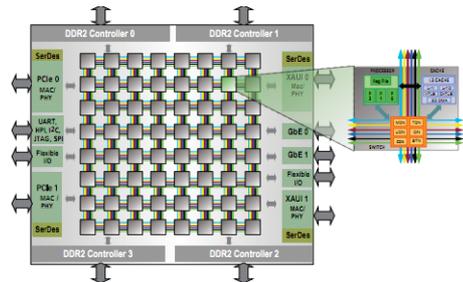
VLIW: very long instruction word
 BOPS: billion operations per second
 mPIPE: multicore Programmable Intelligent Packet Engine
 MiCA: Multicore iMesh Coprocessing Accelerator

TILE-Gx36 vs. TILE Pro64



■ TILE-Gx8036

- ❑ 64-bit VLIW processors
- ❑ 32k L1i cache, 32k L1d cache
- ❑ 256k L2 cache per tile
- ❑ Up to 750 billion op/sec
- ❑ Up to 60 Tbps of on-chip mesh interconnect
- ❑ Over 500 Gbps memory bandwidth
- ❑ 1 to 1.5 GHz operating frequency
- ❑ Power consumption: 10 to 55W for typical applications
- ❑ Two DDR3 memory controllers



■ TILE Pro64

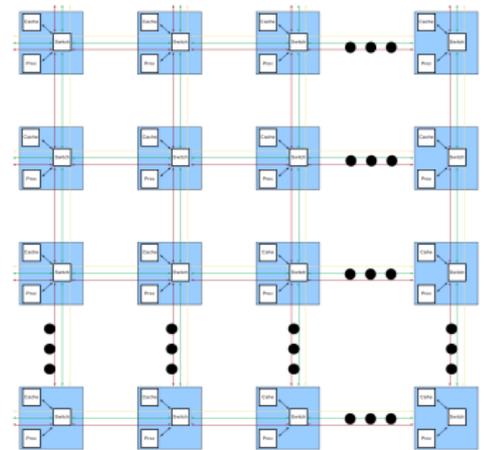
- ❑ 32-bit VLIW processors
- ❑ 16k L1i cache, 8k L1d cache
- ❑ 64k L2 cache per tile
- ❑ Up to 443 billion op/sec
- ❑ Up to 37 Tbps of on-chip interconnect
- ❑ Up to 50 Gbps of I/O bandwidth
- ❑ 700 MHz and 866 MHz operating frequency
- ❑ Power consumption: 19 – 23W @700MHz under full load
- ❑ Four DDR2 memory controllers

iMesh On-Chip Network Fabric

TILEPro64: MDN (Memory), CDN (Coherence), IDN (I/O), TDN (Tile), UDN (User), STN (Static Network)

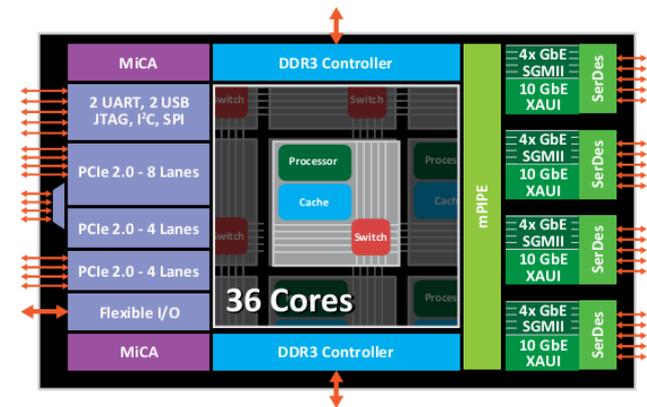
TILE-Gx36: Five networks instead of six (removed static network)

- MDN replaced by QDN (reQuest) and RDN (Response) dynamic network
 - Memory controller has two QDN (0/1) and RDN (0/1) network connections
 - Reduced congestion in memory accesses from *TILEPro*
- SDN (Share)
 - Provides accesses and coherency for cache system
- IDN (Internal)
 - Used primarily for communication with external I/O
- UDN (User)
 - Allows user applications to send data directly between tiles
 - UDN and IDN can be accessed by users



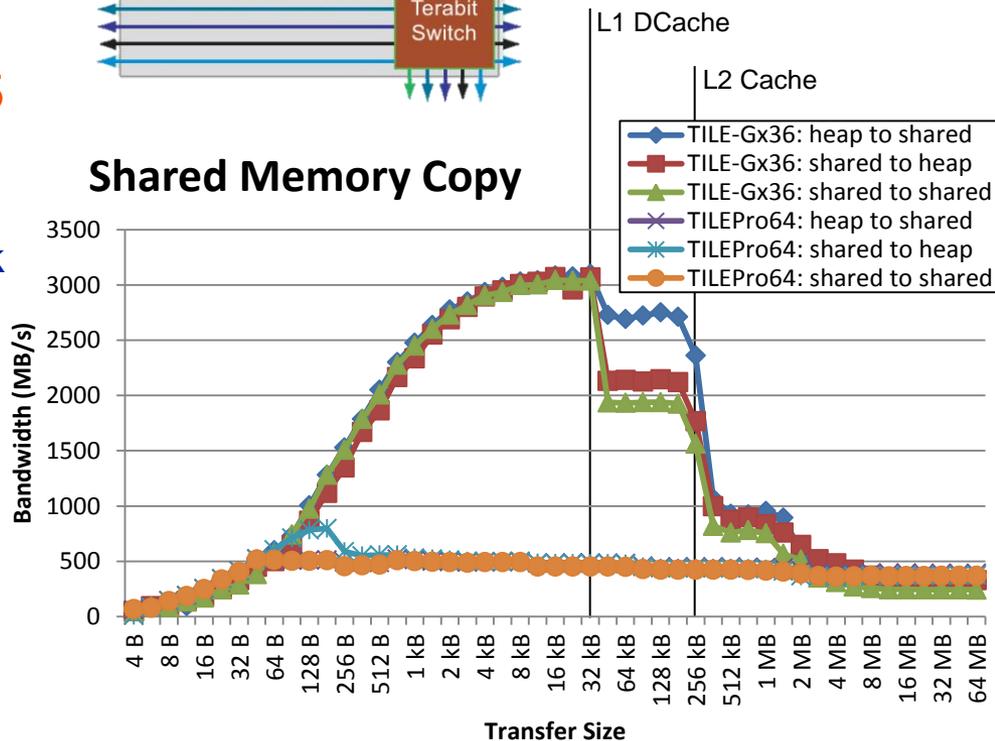
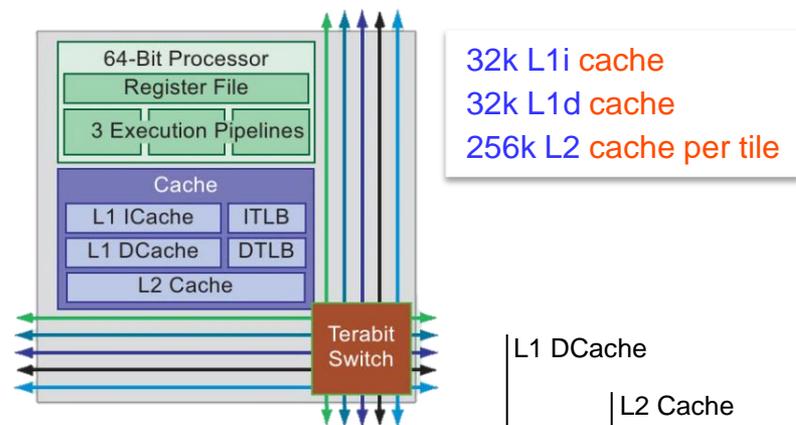
TILE-Gx36 Benchmarking

- Why microbenchmark?
 - Determines *empirically realizable* TILE-Gx performance
 - Defines *realistic upper bound* for TSHMEM performance
- Benchmarking Overview
 - Bandwidth of shared memory copy
 - UDN latency varying test pairs of sender/receiver tiles
 - TMC spin/sync barrier primitives



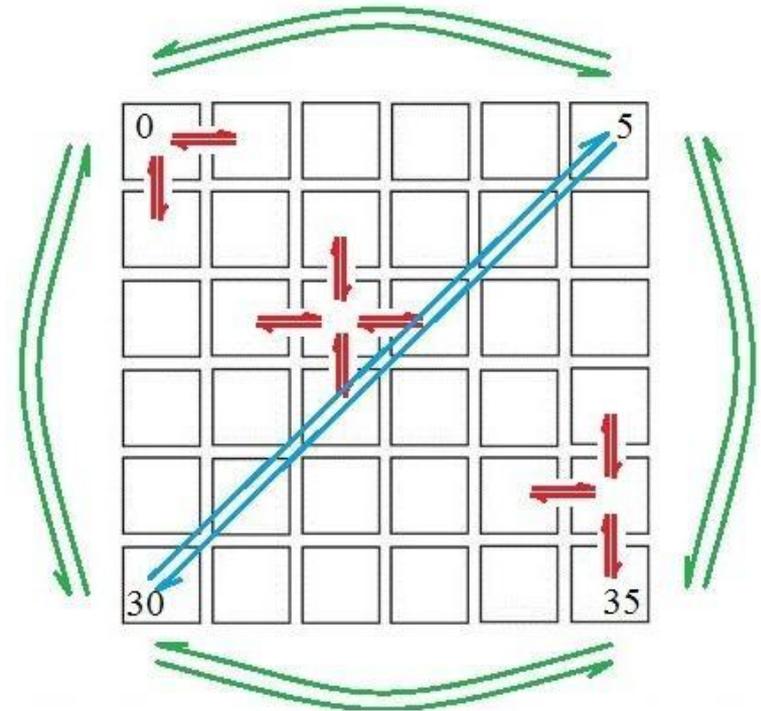
Shared-Memory-Copy Bandwidth

- Bandwidth on iMesh networks to caches and memory controllers
 - Shared memory performance critical for TSHMEM
 - Bandwidth of memory operations influenced by 3 of 5 iMesh networks
 - QDN: memory request network
 - RDN: memory response network
 - SDN: cache sharing network
 - Significant bandwidth performance transitions occur at cache-size limits
 - L1 data cache: **3100 MB/s**
 - L2 cache: **2700 MB/s**
 - Memory-to-memory performance thereafter



UDN Performance on Gx36

- Tiler User Dynamic Network (UDN)
 - Hardware support for routing of data packets between CPU cores (tiles)
- UDN microbenchmark
 - Measures average latency between two tiles by ping-ponging numerous packets



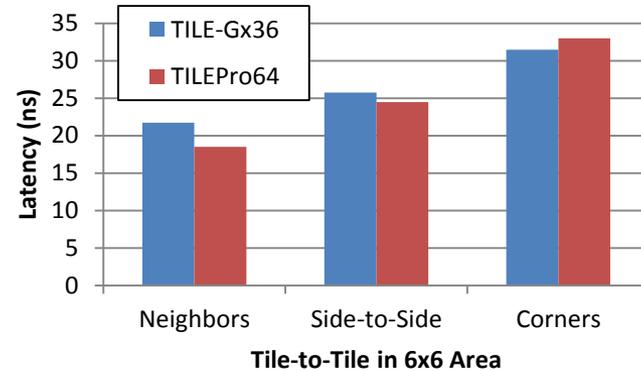
UDN benchmarks transfer between two tiles
Red for neighbors
Green for side to side
Blue for corner to corner

UDN Latency

- Average latency of one-way UDN transfers
 - *TILEPro64's* slightly faster latency is attributed to 32-bit switching fabric
 - TILE-Gx uses 64-bit fabric; yields roughly twice as much data throughput over *TILEPro*
 - Low latency of UDN attractive for fast inter-tile communication



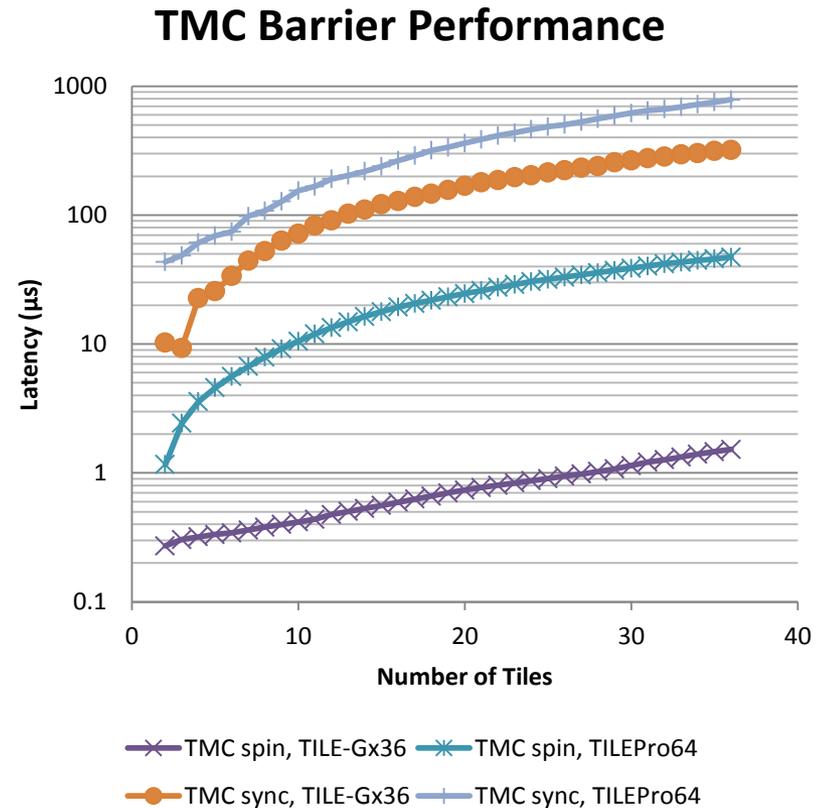
UDN One-Way Average Latency



Type (6x6 area)	Direction	Sender	Receiver	Time (ns)	
				TILE-Gx36	TILEPro64
Neighbors	left	14	13	21	19
	right	14	15	22	19
	up	14	8	22	18
	down	14	20	22	18
	left	28	27	21	19
	right	28	29	22	19
	up	28	22	22	18
	down	28	34	22	18
Side-to-Side	right	6	11	26	25
	left	11	6	25	25
	down	1	31	26	24
	up	31	1	26	24
	right	23	18	25	25
	left	18	23	26	25
	down	33	3	26	24
	up	3	33	26	24
Corners	down-right	0	35	32	33
	up-left	35	0	31	33
	down-left	5	30	31	33
	up-right	30	5	32	33

TMC Spin/Sync Barriers

- Barrier primitives provided by Tiler's TMC library
 - Spin uses spinlock, discouraging process context switching
 - Sync barrier rectifies this with performance penalty
 - Barrier performance on *TILEPro* is order of magnitude slower than *TILE-Gx*
 - *TILEPro* barriers potentially too slow for use in SHMEM

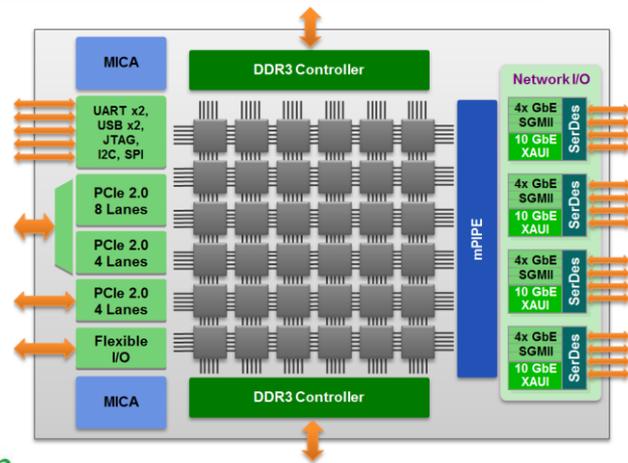
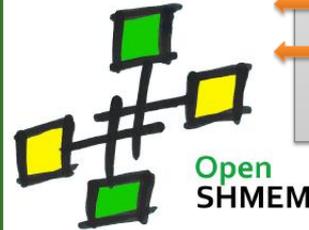


TSHMEM Overview

HPC acceleration with SHMEM on many-core processors

Investigate SHMEM reference design directly over *Tilera TILE-Gx* architecture and libraries

- Stay true to SHMEM principles
 - High performance with low overhead
 - Portability with *easy programmability*
- Maximize architectural benefits
 - Tile interconnect, mPIPE, MiCA
- Extend design to multi-device systems
 - Evaluate interconnect capabilities
 - Explore optimizations to point-to-point transfers and collectives



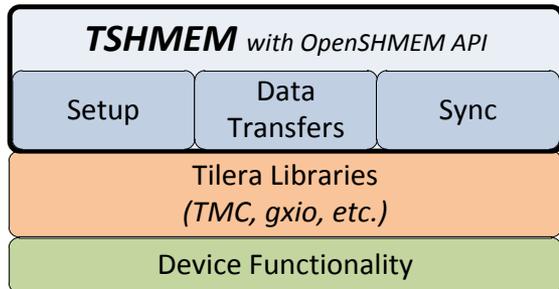
OpenSHMEM and TSHMEM

Achieved

- Dynamic symmetric heap management
- Point-to-point data transfer
- Point-to-point synchronization
- Barrier synchronization
- Broadcast, Collection, Reduction
- Atomics for dynamic variables
- Extension to multiple many-core devices

Ongoing

- Optimizations for multiple many-core
- Exploration of new SHMEM extensions



Modular design utilizing vendor libraries

TSHMEM reference design on TILE-Gx36

TSHMEM Initialization

■ Running TSHMEM applications

- Compile C or C++ program with TSHMEM library



- *tile-gcc* provided as cross compiler for TILE-Gx

- Transfer and run program using *tile-monitor* and *shmemrun* executable launcher

■ TSHMEM setup and initialization – *shmemrun* and *start_pes()*

- Initializes TMC Common Memory (CMEM) for shared memory allocation
- Initializes UDN for barrier sync and basic inter-tile communications
- Creates processes and binds them to unique tiles
- Allocates shared memory and broadcasts pointer
- Synchronizes PEs before exiting routine

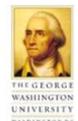
Put/Get Transfers – Design



- Template functions generated with macro definitions
 - Dynamic symmetric variable transfers
 - Performed via *memcpy* and *shmem_ptr* into TMC common memory
- Point-to-point transfers for static symmetric variables
 - Handled via UDN interrupts and dynamic shared memory buffer allocation

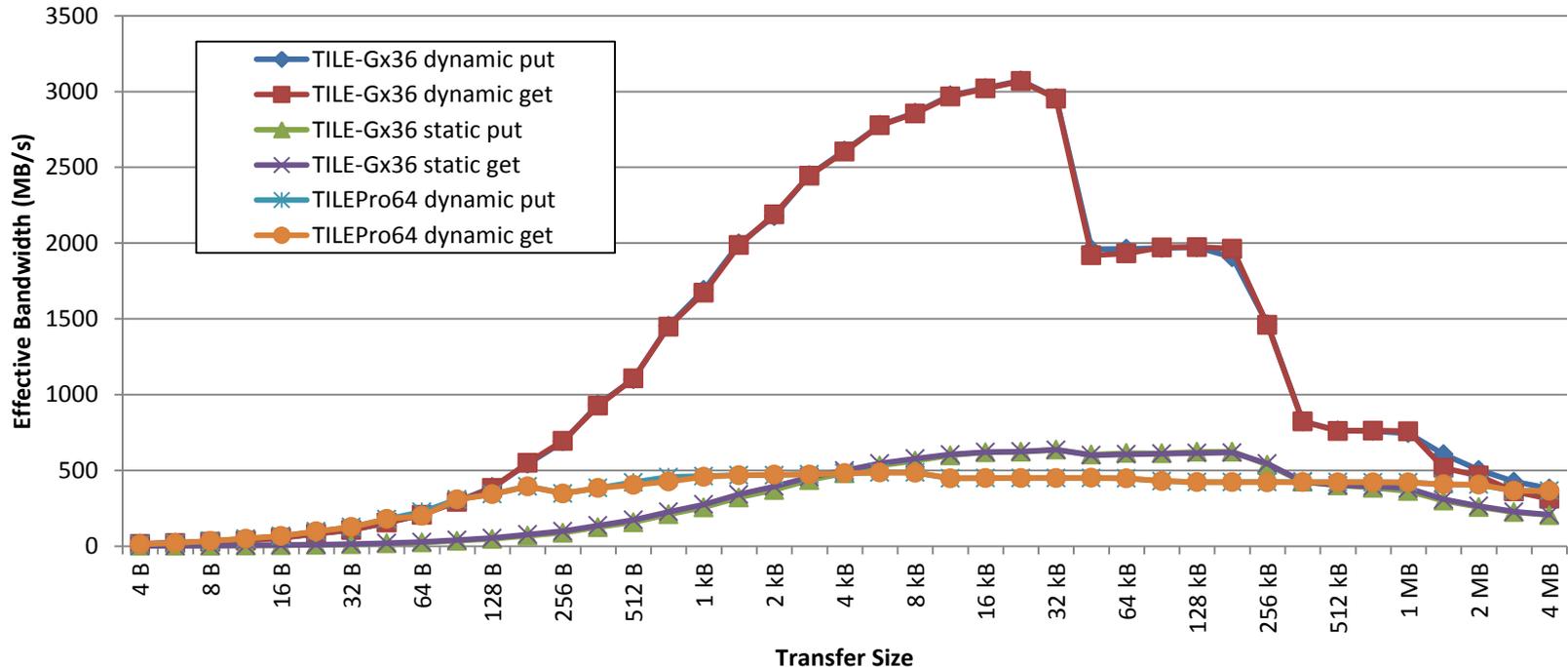
```
23 void
24 shmem_getmem (void *target, const void *source, size_t nelems, int pe)
25 {
26     if (_shmem_ptr_in_my_symheap (source)) {
27         memcpy (target, shmem_ptr ((void*) source, pe), nelems);
28     } else if (shmem_my_pe () == pe) {
29         memcpy (target, source, nelems);
30     } else {
31         _shmem_static_getmem (target, source, nelems, pe);
32     }
33 }
34
35 #define _SHMEM_TYPE_BLOCK_GET( Name, Type )
36
37 void
38 shmem_shmem2z_get (Type *
39 {
40     const size_t bytes
41     shmem_getmem (targ
42 }
43
44
45 #SHMEM_TYPE_BLOCK_GET (cha
46 #SHMEM_TYPE_BLOCK_GET (sho
47 #SHMEM_TYPE_BLOCK_GET (int
48 #SHMEM_TYPE_BLOCK_GET (flo
49 #SHMEM_TYPE_BLOCK_GET (f1o
50 #SHMEM_TYPE_BLOCK_GET (dou
51 #SHMEM_TYPE_BLOCK_GET (lon
52 #SHMEM_TYPE_BLOCK_GET (lon
53
54 #define _SHMEM_SIZE_BLOCK
55
56 void
57 shmem_getmemSize (void *tar
58 {
59     const size_t bytes
60     shmem_getmem (targ
61 }
62
63 #SHMEM_SIZE_BLOCK_GET (sho
64 #SHMEM_SIZE_BLOCK_GET (lon
65 #SHMEM_SIZE_BLOCK_GET (lon
66 #SHMEM_SIZE_BLOCK_GET (lon
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

SHMEM Get



Put/Get Transfers – Performance

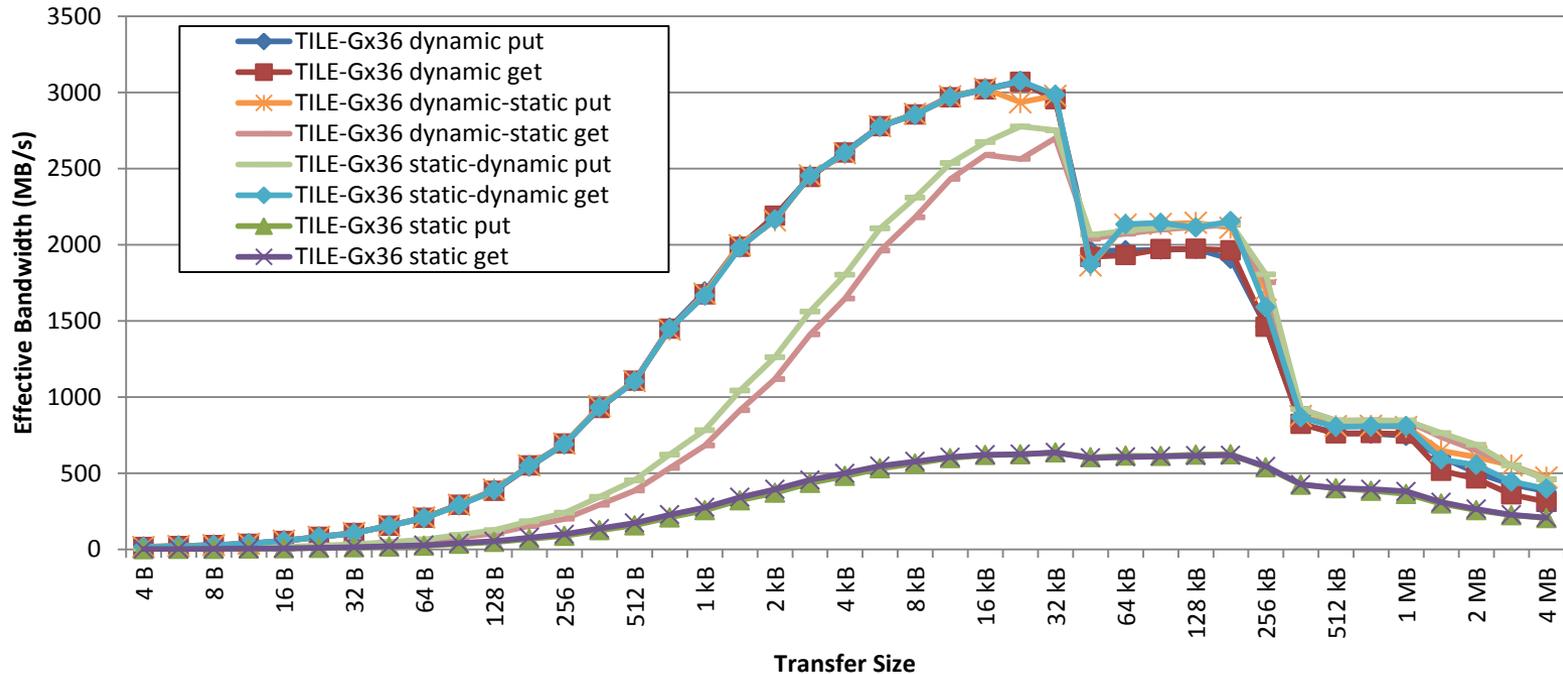
TSHMEM Put/Get Performance



- TSHMEM dynamic performance *closely matches shared-to-shared performance* profile for both TILE-Gx36 and TILEPro64 devices

Static Put/Get Transfers – Perf.

TSHMEM Put/Get Performance on TILE-Gx36



- ❑ TSHMEM static transfers attempt to **offload request** if possible, otherwise a temporary buffer is necessary and drastically degrades performance

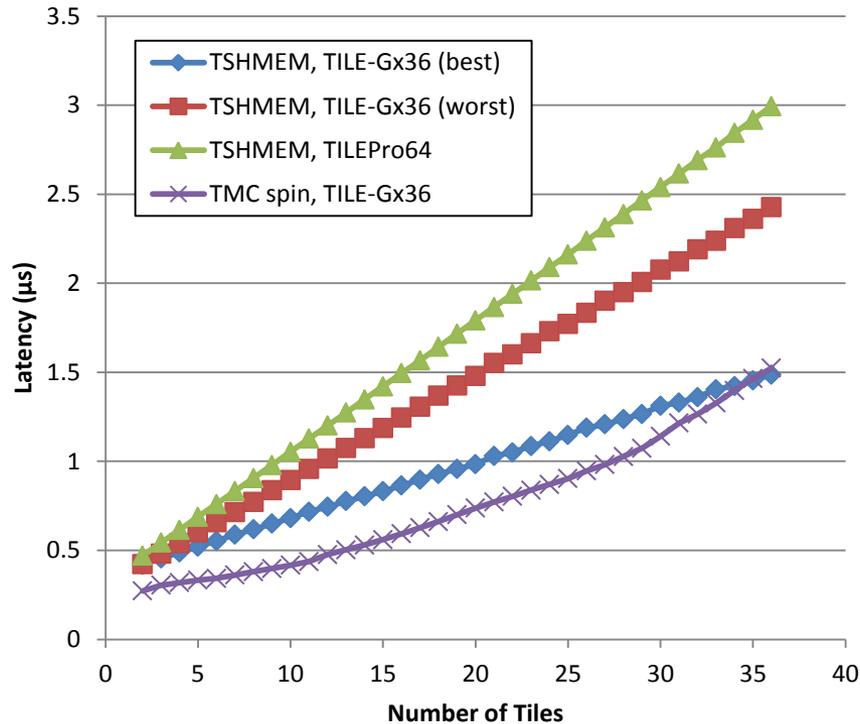
Barrier Sync. – Design



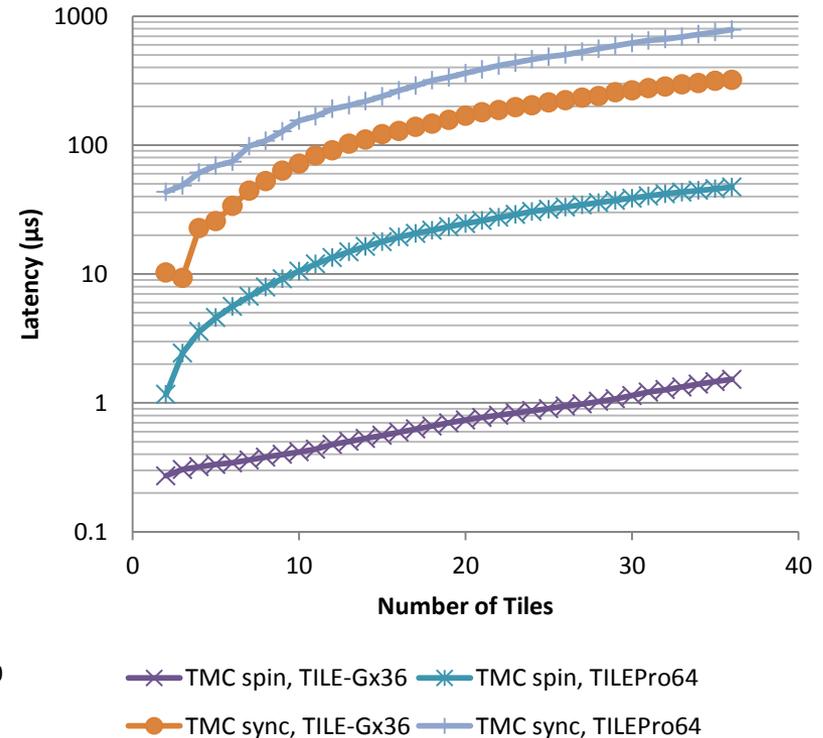
- Barrier uses UDN for messages
 - SHMEM allows for barriers with overlapping active sets
 - Difficult to enable overlapping active sets with TMC barriers
 - Barrier primitives for TILEPro too slow
- Barrier design leverages split-phase barrier internally but externally appears as blocking
 - Semantics for shmem_barrier is blocking
 - Cores (tiles) linearly forward notify message until all receive it
 - Active-set information transmitted to avoid overlap errors from multiple barriers
 - Lead tile broadcasts release message to leave barrier routine
 - Significant performance improvements with increasing number of tiles on TILE-Gx
 - 1.5 μ s latency at 36 tiles

Performance – Barrier Sync

TSHMEM Barrier Performance



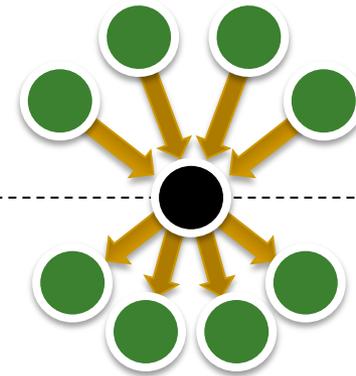
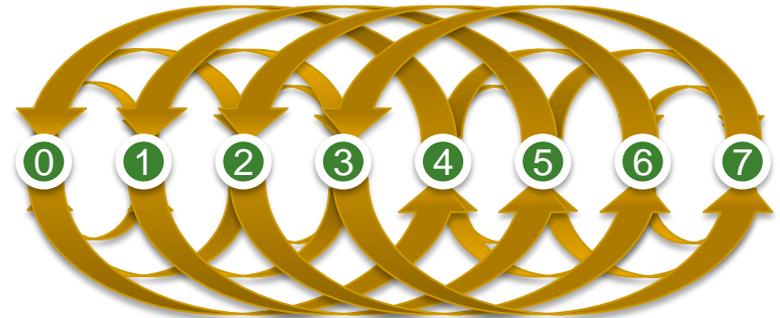
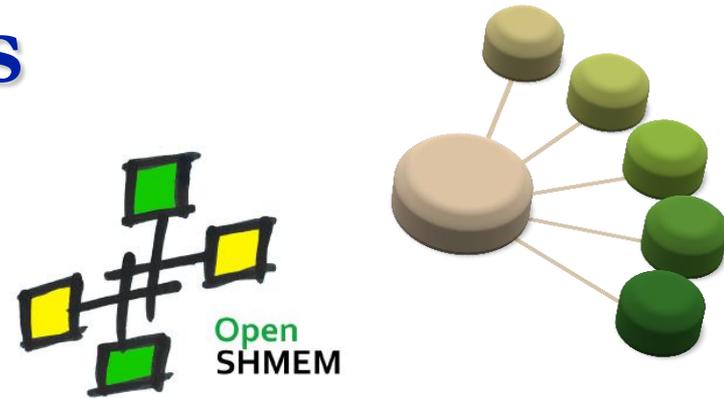
TMC Barrier Performance



- ❑ TSHMEM barriers leverage UDN for **better scaling** than most Tileria TMC barriers for TILE-Gx36 and TILEPro64

Collectives – Types

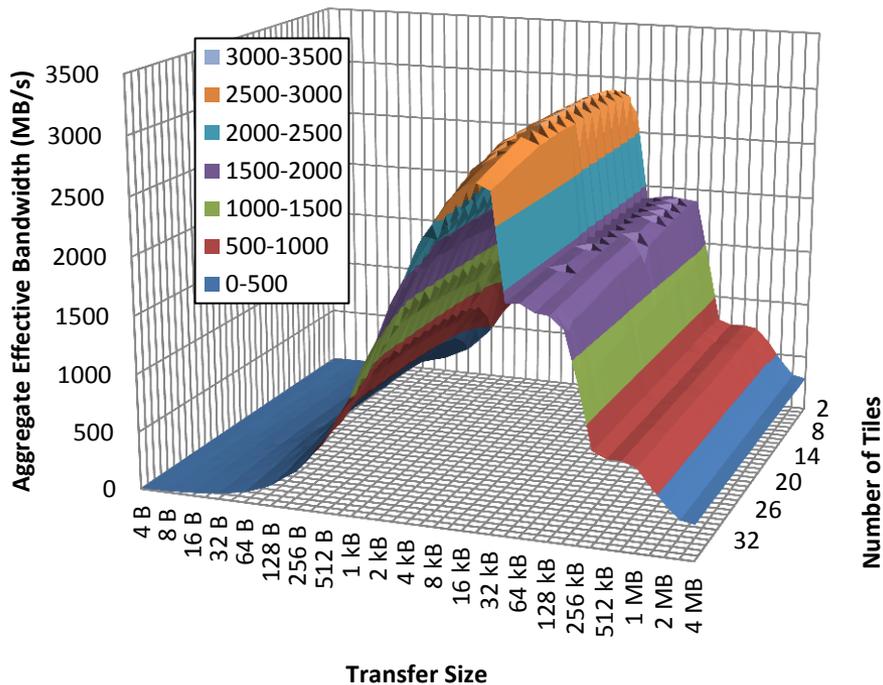
- Broadcast
 - One-to-all transfer
 - Active-set tiles except root tile receive data from root tile's memory
- Collection
 - All-to-all transfer
 - Active-set tiles linearly concatenate their data
- Reduction
 - All-to-all transfer
 - Active-set tiles perform reduction arithmetic operation on their data



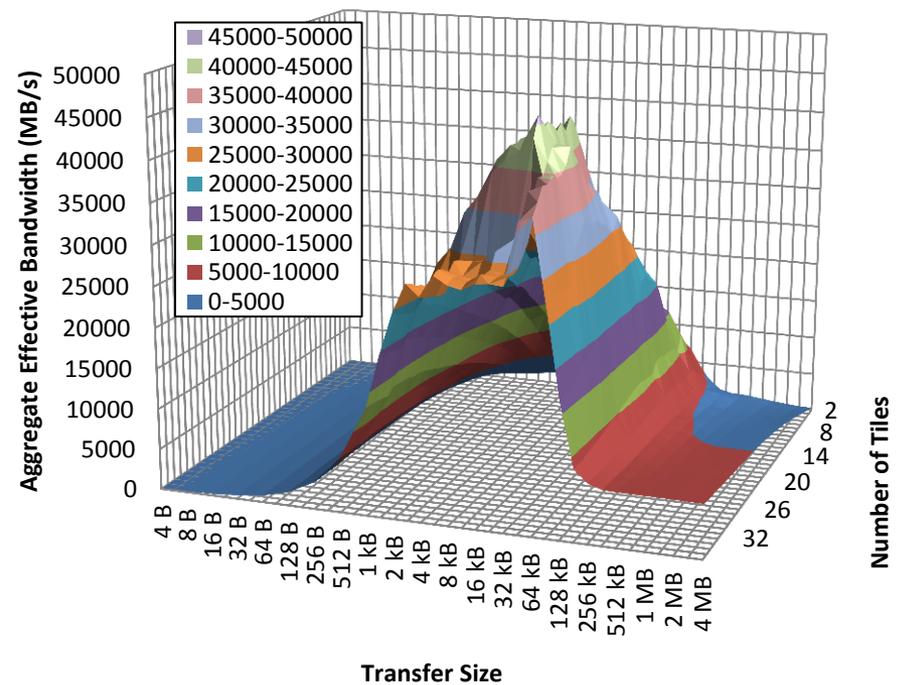
Performance – Broadcast

- Pull-based broadcast up to **46 GB/s aggregate bandwidth** at 29 tiles; 37 GB/s aggregate at 36 tiles

Push-based Broadcast on TILE-Gx36



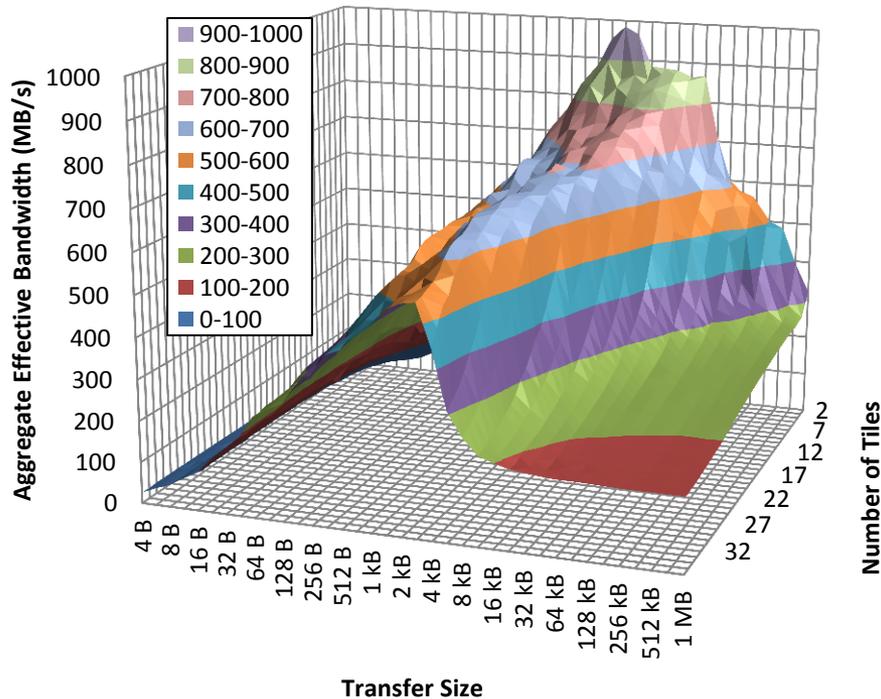
Pull-based Broadcast on TILE-Gx36



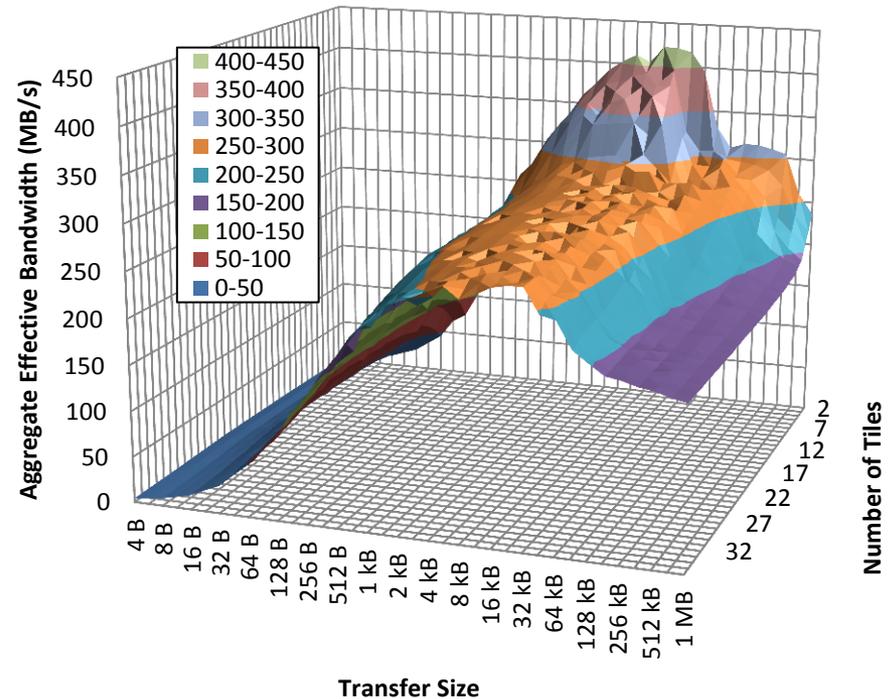
Performance – FCollect, Reduction

- Results are for naïve fast-collect and reduction; additional algorithms planned for testing

Fast-Collect on TILE-Gx36



Reduction on TILE-Gx36



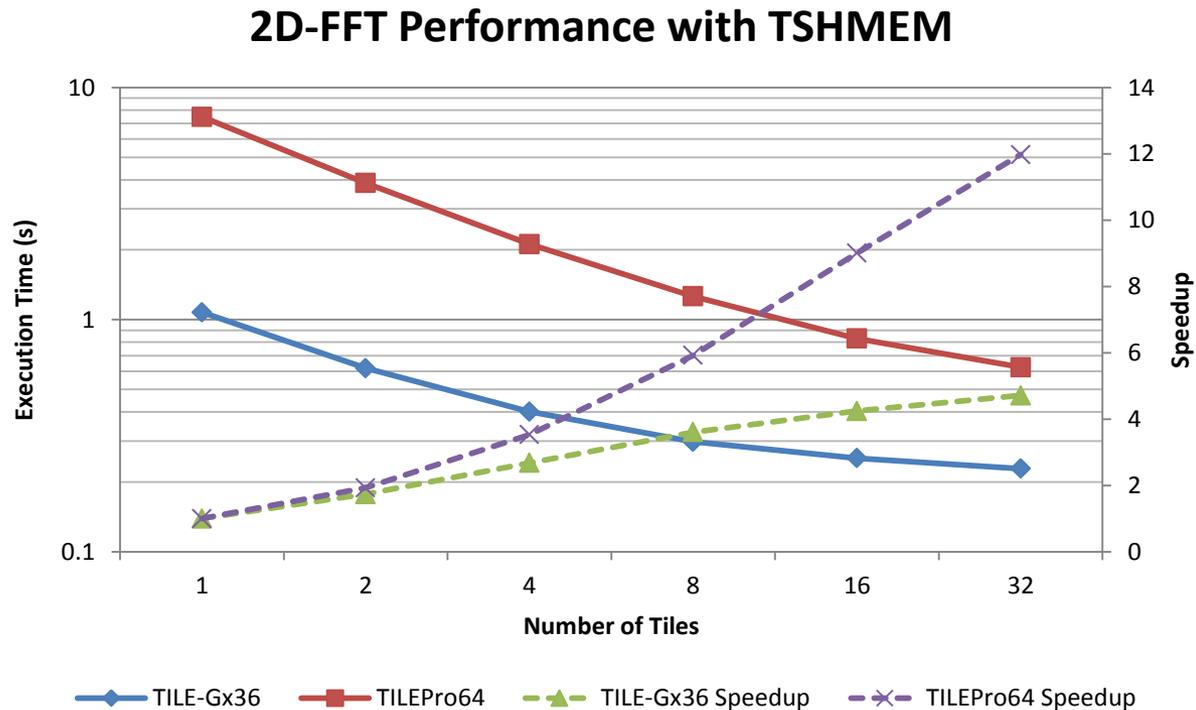
App Case – 2D-FFT

- TILE-Gx executes order of magnitude faster than TILEPro due to *improved floating-point support*

- Speedup at 32

- TILE-Gx: 5

- TILEPro: 12



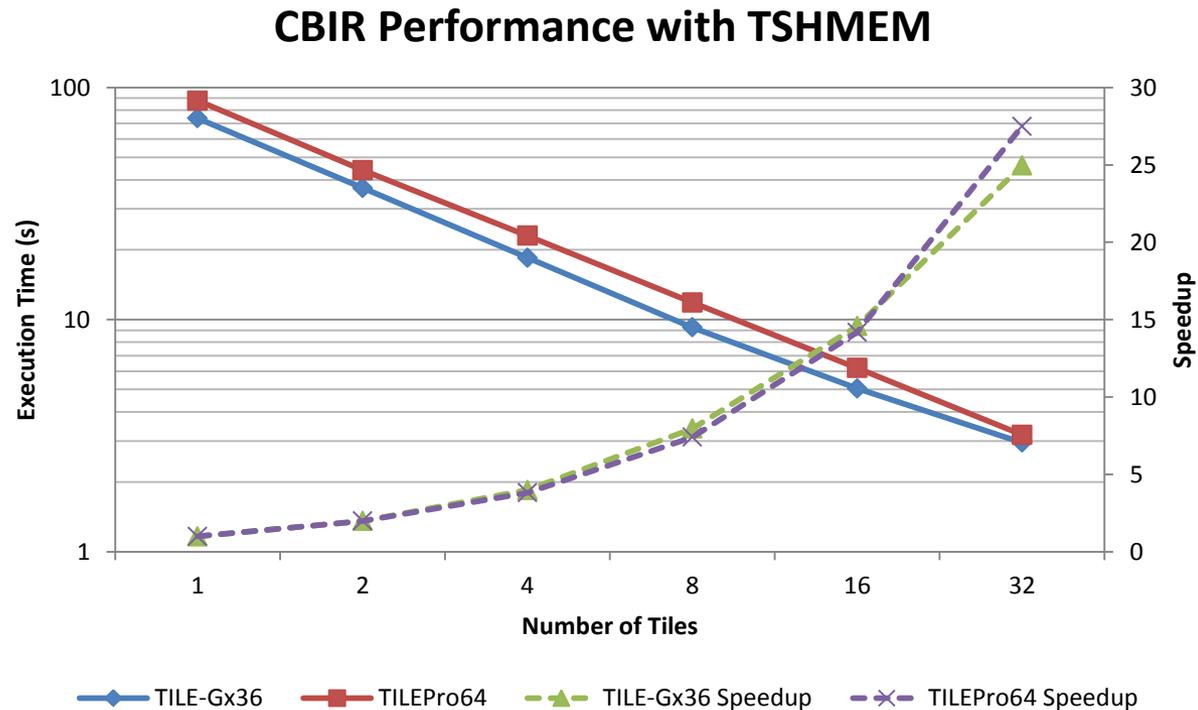
App Case – CBIR

- **Content-based image retrieval** with searches based on feature extraction of input image

- Speedup at 32

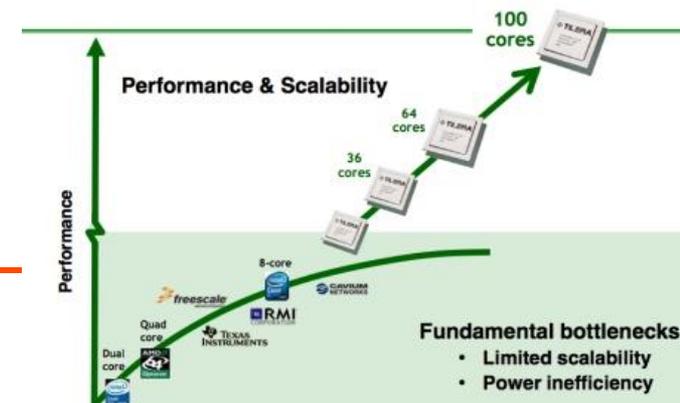
□ TILE-Gx: 25

□ TILEPro: 27



Conclusions

- ***TSHMEM is a new OpenSHMEM library focused on fully leveraging Tileria many-core devices***
- Microbenchmarks define the expected and maximum realizable device performance
 - TSHMEM reaches toward these microbenchmark results with ***very little overhead*** for variety of functions
 - UDN leveraged for SHMEM barriers due to lacking performance of TMC barrier primitives on *TILEPro*
- Performance, portability, scalability demonstrated via two app case studies



Thanks for listening!



Questions?

References

1. OpenSHMEM, “OpenSHMEM API, v1.0 final,” 2012. [Online]. Available: <http://www.openshmem.org/>
2. B. C. Lam, A. D. George, and H. Lam, “An introduction to TSHMEM for shared-memory parallel computing on Tiler many-core processors,” in Proceedings of the 6th Conference on Partitioned Global Address Space Programming Models, ser. PGAS ’12. ACM, 2012.
3. Mellanox Technologies, “Mellanox ScalableSHMEM,” Sunnyvale, CA, USA, 2012. [Online]. Available: [http://www.mellanox.com/related-docs/prod software/PB ScalableSHMEM.pdf](http://www.mellanox.com/related-docs/prod%20software/PB%20ScalableSHMEM.pdf)
4. C. Yoon, V. Aggarwal, V. Hajare, A. D. George, and M. Billingsley, III, “GSHMEM, a portable library for lightweight, shared-memory, parallel programming,” in Proceedings of the 5th Conference on Partitioned Global Address Space Programming Models, ser. PGAS ’11. ACM, 2011.
5. D. Bonachea, “GASNet specification, v1.1,” University of California at Berkeley, Berkeley, CA, USA, Tech. Rep., 2002.
6. Tiler Corporation, “TILE-Gx8036 processor specification brief,” San Jose, CA, USA, 2012. [Online]. Available: [http://www.tiler.com/sites/default/files/productbriefs/TILE-Gx8036 PB033-02.pdf](http://www.tiler.com/sites/default/files/productbriefs/TILE-Gx8036PB033-02.pdf)
7. ———, “TILEPro64 processor product brief,” San Jose, CA, USA, 2011. [Online]. Available: [http://www.tiler.com/sites/default/files/productbriefs/TILEPro64 Processor PB019 v4.pdf](http://www.tiler.com/sites/default/files/productbriefs/TILEPro64%20Processor%20PB019%20v4.pdf)
8. J. Huang, S. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, “Image indexing using color correlograms,” in Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on, Jun 1997, pp. 762–768.