Shared memory π-calculation & The STREAM benchmark

How this presentation is organized

Presenting shared memory π-calculation:

- Objective
- Requirements
- Code and scaling study
- Conclusion

STREAM benchmark:

- Flat vs Cache mode
- Quadrant vs SNC4

Shared memory
π-calculation

Objective

The value of π can be approximated, by means of integrating the function

$$\varphi(x) = \frac{1}{1 + x^2}$$

over the interval [0, 1].

- The aim of this task was to develop a serial implementation of this integration, and to parallelize the application using OpenMP.

Requirements

- Develop a serial implementation that integrates function ϕ (x) over [0, 1].
- Parallelize your application using OpenMP
  - critical directive.
  - reduction clause.
- Perform a scaling study of your algorithm.
  - OMP_NUM_THREADS
  - Weak scaling
  - Strong scaling

Mathematical background

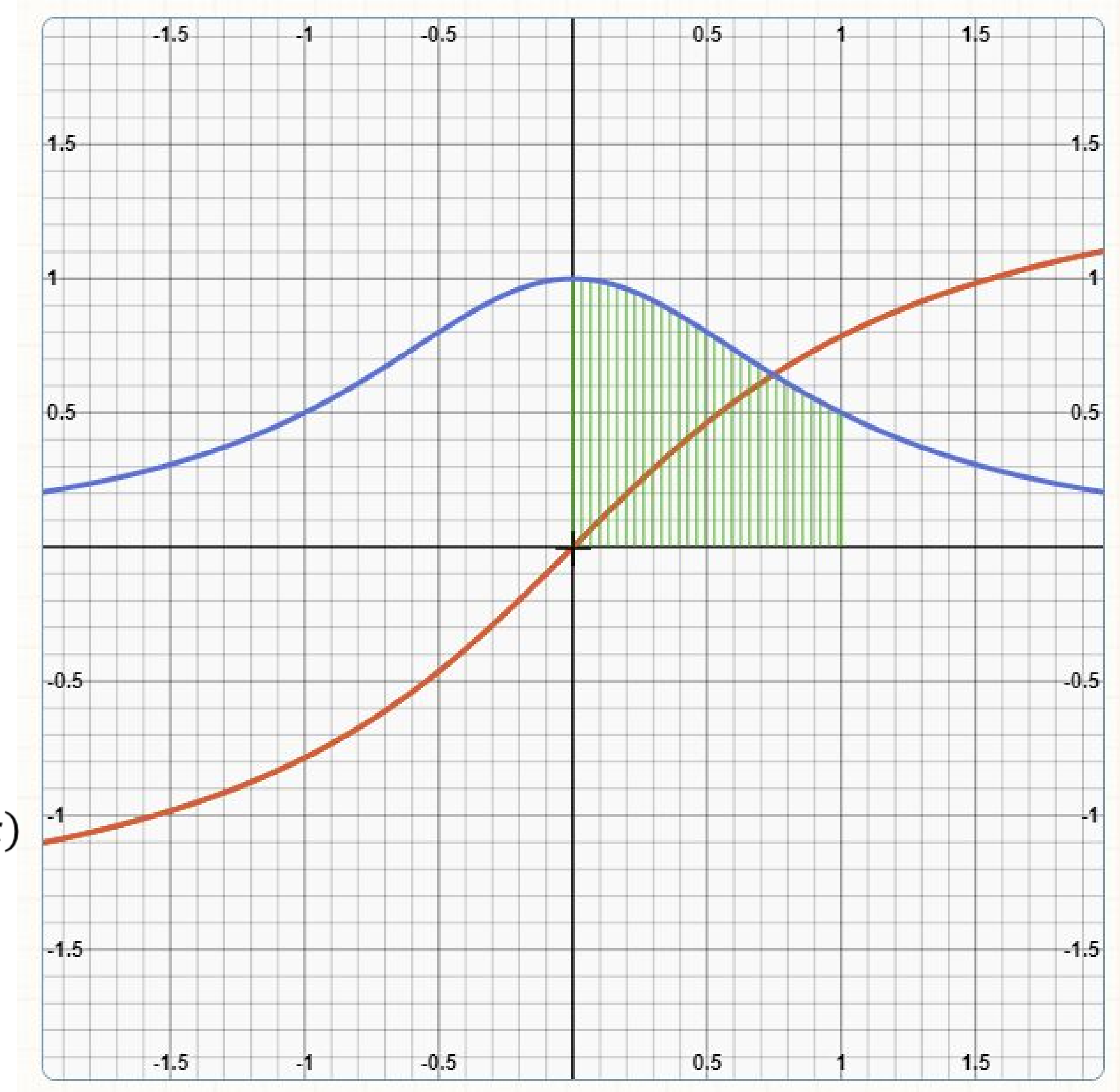
$$\varphi(x) = \frac{1}{1 + x^2}$$

$$\int \varphi(x)\, dx = \arctan(x)$$

$$\int_0^1 \varphi(x)\, dx = \arctan(1) - \arctan(0) = \frac{\pi}{4}$$

=> If we multiply the result of integration by 4, we get the value of π.

Graph:



Code

```
double h, y, sum;
//number of partitions
long n = 1000000000;

h = 1. / n;
sum = 0;

for (i = 0; i <= n; i++) {
    //calculate function value at current partition
    y = phi(i*h);
    //add current function value to sum
    sum += y;
}

sum *= 4. * h; //value of pi
```

The critical directive

- Specifies a region of code that must be executed by only one thread at a time.

```
#pragma omp parallel for private(y), shared(sum)
    for (i = 0; i <= n; i++)  {
        y = phi(i*h);

#pragma omp critical
        sum += y;
    }
```

Scaling study – Critical directive

- Weak scaling:
  - n = 25 000 000

N/No. of threads	n/16	2n/32	4n/64	8n/128
Execution time (s)	17.931	36.884	76.407	149.951

- Strong scaling:
  - n = 100 000 000.

No. of threads	16	32	64	128
Execution time (s)	73.772	73.902	77.451	78.451

The reduction clause

- Performs a reduction operation on the variables that appear in its list.

```
#pragma omp parallel for private(y), reduction(+: sum)
    for (i = 0; i <= n; i++)  {
        y = phi(i*h);
        sum += y;
    }
```

Scaling study – Reduction clause

- Weak scaling:
  - n = 250 000 000

N/No. of threads	n/16	2n/32	4n/64	8n/128
Execution time (s)	0.176	0.181	0.202	0.302

- Strong scaling:
  - n = 1 000 000 000.

No. of threads	16	32	64	128
Execution time (s)	0.610	0.325	0.196	0.188

Conclusion

- Using the critical directive kills performance because threads have to access the critical section one after another.
- Parallelizing with the reduction clause leads to faster execution once more threads are introduced.

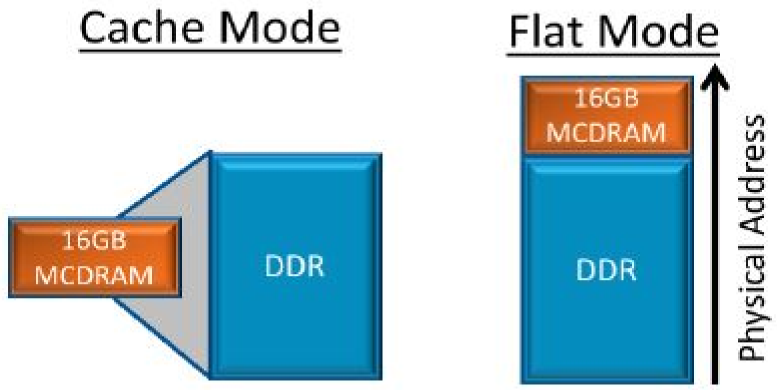
- Strong scaling shows that if the problem is big enough adding more threads to parallelize it works up to a certain point. It will have diminishing returns once the problem chunks are so small that the overhead of adding more threads doesn’t help anymore.

STREAM benchmark

Sub-benchmark types

- Copy - applies c[i] = a[i] on the arrays
- Scale - applies b[i] = scalar*c[i]
- Add - applies c[i] = a[i] + b[i]
- Triad - applies a[i] = b[i] + scalar*c[j]

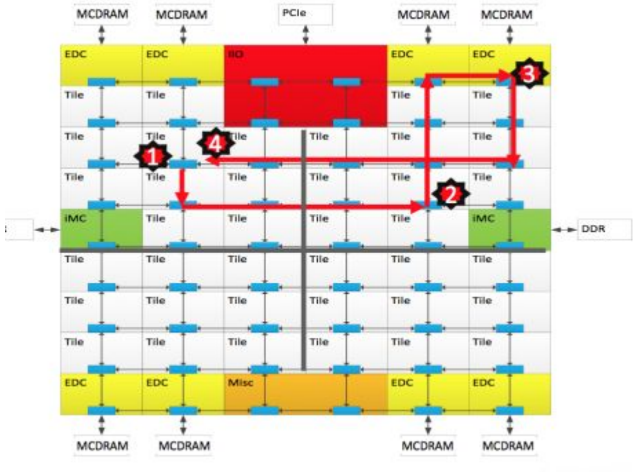
Flat vs Cache mode



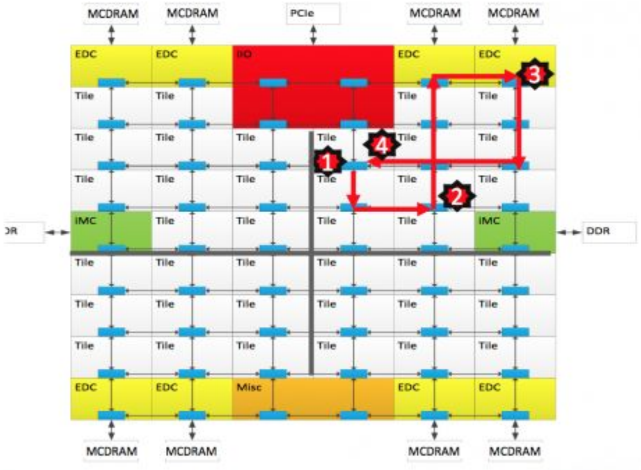
- Cache mode
  - MCDRAM acts as L3 direct mapped cache
  - transparent to the user
  - suitable for legacy applications
  - spatial & temporal locality applications can achieve peak performance
- Flat mode
  - MCDRAM and DDR4 can be allocated selectively (using Memkind library)
  - gives user control over the data that goes into the high bandwidth memory
  - useful if application is limited by DDR4 bandwidth

Quadrant vs SNC4

- Quadrant
  - the chip is divided into 4 virtual quadrants
  - tag directory and memory channel are in the same quadrant
  - transparent to the user (the OS sees the system as one NUMA node)
  - easier to use
  - usually provides good performance (less mesh traffic)
- SNC4
  - each quadrant is exposed as a separate NUMA
  - software must be optimized for NUMA architecture
  - best performance



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Stream tests

- Array size = 80000000 (elements), Offset = 0 (elements)
- Memory per array = 610.4 MiB (= 0.6 GiB).
- Total memory required = 1831.1 MiB (= 1.8 GiB)
- OMP_NUM_THREADS=64

Quadrant

Cache

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	228009.4	0.005730	0.005614	0.005889
Scale:	233666.0	0.005561	0.005478	0.005701
Add:	146352.8	0.013542	0.013119	0.013960
Triad:	266622.4	0.007758	0.007201	0.008317

Flat

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	76583.2	0.017029	0.016714	0.022321
Scale:	76600.6	0.017015	0.016710	0.021970
Add:	82431.5	0.024193	0.023292	0.028823
Triad:	82230.3	0.023611	0.023349	0.027358

SNC4

Cache

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	218988.0	0.005921	0.005845	0.005991
Scale:	217356.6	0.005956	0.005889	0.006097
Add:	185631.5	0.011186	0.010343	0.011798
Triad:	271760.0	0.007150	0.007065	0.007254

Flat

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	12225.0	0.105086	0.104704	0.106260
Scale:	12084.6	0.106379	0.105920	0.107742
Add:	13078.7	0.147689	0.146804	0.152449
Triad:	13013.2	0.148147	0.147543	0.148948

numactl -m 1

Quad

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	221682.6	0.005875	0.005774	0.005953
Scale:	307697.7	0.004223	0.004160	0.004293
Add:	220740.7	0.008891	0.008698	0.009048
Triad:	370306.9	0.005228	0.005185	0.005280

SNC4

Function	Best Rate MB/s	Avg time	Min time	Max time
Copy:	223110.5	0.005863	0.005737	0.005953
Scale:	310815.1	0.004213	0.004118	0.004266
Add:	222196.4	0.008890	0.008641	0.009032
Triad:	369728.8	0.005229	0.005193	0.005274

Questions ?