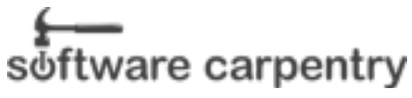
Modern parallelism models

Presented by Alex Kosenkov



Agenda

- PGAS
- Cilk
- TBB
- Dataflow

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

PGAS

Partitioned global address space:

- Unified Parallel C, SHMEM
- Coarray Fortran
- Fortress
- Chapel
- X10
- Global Arrays
- ZPL

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

software carpentry

PGAS: OpenSHMEM

```c
#include <shmem.h>

long pSync[_SHMEM_BARRIER_SYNC_SIZE]; /* Symmetric work array */
int number;                           /* Symmetric data */

int main(){
    start_pes(0);                     /* Initialize the OpenSHMEM library */
    int npes = num_pes();             /* Get the number of PEs */
    int me = my_pe ();                /* Get the rank of the PE */

    if(me == 0){
        shmem_int_put(&number, &number, 1, 1);
        shmem_quiet();
    }
    shmem_barrier(0,0,2,pSync);
    if(me == 1) printf("PE 1 received number %d from PE 0\n", number);

    shmem_barrier_all();              /* Synchronize */
    return (0);
}
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

PGAS: UPC

Example: adding two vectors

```upc
#include <upc_relaxed.h>
#define N 100*THREADS

shared int v1[N], v2[N], v1plusv2[N];

int main(){
    for(int i = MYTHREAD; i < N; i += THREADS) v1plusv2[i]=v1[i]+v2[i];
    // upc_forall(i = 0; i < N; i++; i) v1plusv2[i]=v1[i]+v2[i];
    return 0;
}
```

Example: matrix-vector multiplication

```upc
#include <upc_relaxed.h>

shared [THREADS] int a[THREADS][THREADS];
shared int b[THREADS], c[THREADS];

int main(){
    int i, j;
    upc_forall(i = 0; i < THREADS; i++; i){
        c[i] = 0;
        for(j = 0; j < THREADS; j++)
            c[i] += a[i][j]*b[j];
    }
    return 0;
}
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

PGAS: Chapel

Example: tree traversal

```
 1 config const treeHeight: uint = 4;
 2
 3 class node {
 4   var id: int;
 5   var left, right: node;
 6   proc ~node() {
 7     if left then delete left;
 8     if right then delete right;
 9   }
10 }
11
12 proc buildTree(height: uint = treeHeight, id: int = 1): node {
13   var newNode = new node(id);
14
15   if height > 1 {
16     cobegin {
17       newNode.left  = buildTree(height-1, id + 1);
18       newNode.right = buildTree(height-1, id + (1 << (height-1)));
19     }
20   }
21   return newNode;
22 }
23
24 proc sum(n: node): int {
25   var total = n.id;
26
27   if n.left != nil {
28     var sumLeft, sumRight: int;
29     cobegin ref(sumLeft, sumRight) {
30       sumLeft = sum(n.left);
31       sumRight = sum(n.right);
32     }
33     total += (sumLeft + sumRight);
34   }
35   return total;
36 }
37
38 proc main() {
39   var root = buildTree();
40   writeln("sum=", sum(root));
41   delete root;
42 }
```

Example: distributed linked list

```
 1 class Node {
 2   var data: real;
 3   var next: Node;
 4 }
 5
 6 var head = new Node(0);
 7 var current = head;
 8 for i in 1..numLocales-1 do
 9   on Locales[i] {
10     current.next = new Node(i);
11     current = current.next;
12   }
13
14 current = head;
15 while current {
16   on current {
17     writeln("node with data = ", current.data, " on locale ", here.id);
18     current = current.next;
19   }
20 }
```

Task Parallelism:

```
cobegin { task1(); task2(); }
coforall i in aggregate do task(i);
sync { begin task1(); begin task2(); }
```

Data Parallelism:

```
forall a in A do a = 1.0;
[a in A] a = 1.0;
A = 1.0;
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Intel Cilk Plus

Task-based parallelism in three keywords:

- cilk_for
- cilk_spawn
- cilk_sync

```
~/Dropbox/sandbox $ module load gcc/4.9.0
~/Dropbox/sandbox $ g++ cilk.cpp -fcilkplus
~/Dropbox/sandbox $ icpc cilk.cpp
~/Dropbox/sandbox $
~/Dropbox/sandbox $ CILK_NWORKERS=12 ./a.out
```

Example: matrix-vector multiplication

```
      ...
18    cilk_for(int i = 0; i < N; i++)
19    for(int j = 0; j < N; j++)
20    c[i] += a[i+j*N]*b[j];
      ...
```

Example: the Fibonacci numbers

```
 1 #include <stdio.h>
 2 #include <cilk/cilk.h>
 3
 4 int fib(int n){
 5     if (n < 2) return n;
 6     int x = cilk_spawn fib(n-1);
 7     int y = fib(n-2);
 8     cilk_sync;
 9     return x + y;
10 }
11
12 int main(){
13     printf("%d\n", fib(40));
14     return 0;
15 }
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Intel Cilk Plus

Extensions for Array Notations Programming Model:

section_operator ::=
[<lower bound>:<length>:<stride>]

Example: vectorisation (saxpy)

```
1 #include <cilk/cilk.h>
2 #include <stdio.h>
3 #include <algorithm>
4
5 void saxpy_vec(int m, float a, float x[m], float y[m]){
6     y[:] += a*x[:];
7 }
8
9 int main(){
10     float a[2048], b[2048];
11     a[:] = b[:] = std::rand();
12     cilk_for (int i = 0; i < 2048; i +=256){
13         saxpy_vec(256, 2.0, &a[i], &b[i]);
14     }
15     return 0;
16 }
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Intel Threading Building Blocks

- Algorithms — tbb::parallel_for, tbb::parallel_reduce, tbb::parallel_sort
- Containers — tbb::concurrent_vector
- Allocators — tbb::cache_aligned_allocator, tbb::scalable_allocator
- Atomics — tbb::fetch_and_add, tbb::compare_and_swap
- Mutexes — tbb::mutex, tbb::spin_mutex

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Intel Threading Building Blocks

Example: finding average + reduction

```
1 #include "tbb/parallel_for.h"
2 #include "tbb/parallel_reduce.h"
3 #include "tbb/blocked_range.h"
4 #include <algorithm>
5 #include <iostream>
6 #define N 5000
7
8 void average(float* in, float* out, int size){
9     static tbb::affinity_partitioner p;
10    tbb::parallel_for( tbb::blocked_range<int>( 1, N ), [&](const tbb::blocked_range<int>& range){
11        for(int i = range.begin(); i != range.end(); ++i)
12        out[i] = (in[i-1]+in[i]+in[i+1])*(1/3.f);
13    }, p);
14 }
15
16 float sum(float* array, size_t n){
17    return tbb::parallel_reduce( tbb::blocked_range<float*>( array, array+n ), 0.f,
18                                 [](const tbb::blocked_range<float*>& r, float value) -> float {
19                                     return std::accumulate(r.begin(),r.end(),value);
20                                 }, std::plus<float>());
21 }
22
23 int main(){
24    float out[N];  out[:] = 0.;
25    float in[N+1]; in[:] = 2.;
26    average(in, out, N);
27    std::cout << "Total: " << sum(out,N) << "\n";
28    return 0;
29 }
```

```
~/Dropbox/sandbox $ icpc tbb.cpp -std=c++11 -ltbb \
                    -I/apps/castor/intel-2013/tbb/include \
                    -L/apps/castor/intel-2013/tbb/lib/intel64
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Intel Threading Building Blocks

Example: reduction (alternative)

```
1  #include "tbb/parallel_for.h"
2  #include "tbb/parallel_reduce.h"
3  #include "tbb/blocked_range.h"
4
5  float sum_v2(const float a[], size_t n){
6      class Reduce {
7          public:
8          void operator()( const tbb::blocked_range<size_t>& r ) {
9              sum_l += __sec_reduce_add(part[r.begin():r.size()]);
10         }
11         void join(const Reduce& y){ sum_l += y.sum_l; }
12
13         Reduce(Reduce& x, tbb::split) : part(x.part), sum_l(0) {}
14         Reduce(const float* a) : part(a), sum_l(0) {}
15     public:
16         float sum_l;
17     private:
18         const float* part;
19     };
20
21     Reduce sf(a);
22     tbb::parallel_reduce( tbb::blocked_range<size_t>(0,n), sf );
23     return sf.sum_l;
24 }
```

Dataflow

- Join-pattern (Join Java, Parallel C#, T++)
- Ambient
- Linda (similar to Charm++/E)
- CAL
- VHDL
- Verilog

Dataflow: Ambient

Example: plus/mul for vector elements

```
1 #include "ambient/ambient.hpp"
2 #include "ambient/container/vector.hpp"
3 #define N 1000
4
5 int main(){
6
7     ambient::vector<int> a(N, 13);
8     ambient::vector<int> b(N, 10);
9     {
10        ambient::actor select(ambient::scope::begin());
11        a += b;
12    }
13    {
14        ambient::actor select(ambient::scope::begin()+1);
15        ambient::async([&](ambient::vector<int>& a, const ambient::vector<int>& b){
16                versioned(a).data[0:get_length(a)] *= versioned(b).data[0:get_length(a)];
17        }, a, b);
18    }
19    ambient::sync();
20    return 0;
21 }
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Dataflow: Ambient

Example: Strassen's Matrix Multiplication

```
template<class MatrixA, class MatrixB, class Matrix>
void gemm_strassen(tiles<MatrixA>&& a, tiles<MatrixB>&& b, tiles<Matrix>&& c){
    size_t n  = c.cols/2;
    size_t nt = c.nt/2;
    if(nt){
        tiles<matrix<value_type,allocator_type> > m1(n, n);
        tiles<matrix<value_type,allocator_type> > m2(n, n);
        tiles<matrix<value_type,allocator_type> > m3(n, n);
        tiles<matrix<value_type,allocator_type> > m4(n, n); m4 = a.subset(0, 0, nt, nt);
        tiles<matrix<value_type,allocator_type> > m5(n, n); m5 = b.subset(0, 0, nt, nt);
        tiles<matrix<value_type,allocator_type> > d(n*2, n*2); d = a;
        tiles<matrix<value_type,allocator_type> > e(n*2, n*2); e = b;

        d.subset(0,  0,  nt, nt) += a.subset(0,  nt, nt, nt);  e.subset(0,  0,  nt, nt) += b.subset(0,  nt, nt, nt);
        d.subset(0,  nt, nt, nt) -= a.subset(nt, nt, nt, nt);  e.subset(0,  nt, nt, nt) -= b.subset(nt, nt, nt, nt);
        d.subset(nt, nt, nt, nt) += a.subset(nt, 0,  nt, nt);  e.subset(nt, nt, nt, nt) += b.subset(nt, 0,  nt, nt);
        d.subset(nt, 0,  nt, nt) -= a.subset(0,  0,  nt, nt);  e.subset(nt, 0,  nt, nt) -= b.subset(0,  0,  nt, nt);

        m4 += a.subset(nt, nt, nt, nt);  m5 += b.subset(nt, nt, nt, nt);

        gemm_strassen(d.subset(0,  nt, nt, nt),
                      e.subset(nt, nt, nt, nt),
                      c.subset(0,  0,  nt, nt));
        gemm_strassen(a.subset(0,  0,  nt, nt),
                      e.subset(0,  nt, nt, nt),
                      c.subset(0,  nt, nt, nt));
        gemm_strassen(d.subset(nt, nt, nt, nt),
                      b.subset(0,  0,  nt, nt),
                      c.subset(nt, 0,  nt, nt));
        gemm_strassen(d.subset(nt, 0,  nt, nt),
                      e.subset(0,  0,  nt, nt),
                      c.subset(nt, nt, nt, nt));

        gemm_strassen(std::move(m4), std::move(m5),
                      std::move(m1));
        gemm_strassen(d.subset(0,  0,  nt, nt),
                      b.subset(nt, nt, nt, nt),
                      std::move(m2));
        gemm_strassen(a.subset(nt, nt, nt, nt),
                      e.subset(nt, 0,  nt, nt),
                      std::move(m3));

        c.subset(nt, nt, nt, nt) += c.subset(0,  nt, nt, nt);
        c.subset(nt, nt, nt, nt) -= c.subset(nt, 0,  nt, nt);

        c.subset(0,  0,  nt, nt) += m1;
        c.subset(nt, nt, nt, nt) += m1;
        c.subset(0,  nt, nt, nt) += m2;
        c.subset(0,  0,  nt, nt) -= m2;
        c.subset(0,  0,  nt, nt) += m3;
        c.subset(nt, 0,  nt, nt) += m3;
    }else{
        gemm(a[0], b[0], c[0]);
    }
}
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

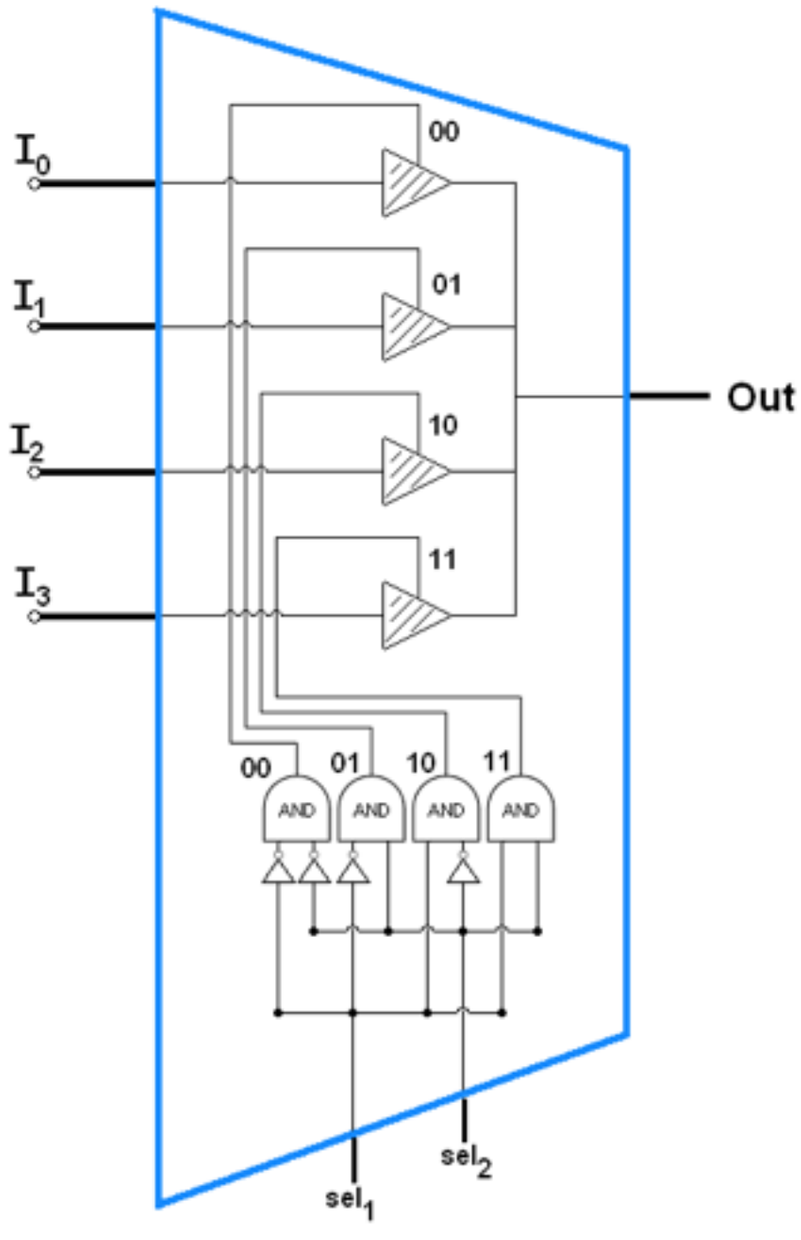
CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Dataflow: VHDL/Verilog

Example: multiplexer in Verilog

```
1 module mux6( select, d, q );
2
3 input[1:0] select;
4 input[3:0] d;
5 output q;
6
7 reg q;
8 wire[1:0] select;
9 wire[3:0] d;
10
11 always @( select or d)
12 begin
13     q =  ( ~select[0] & ~select[1] & d[0] )
14        | (  select[0] & ~select[1] & d[1] )
15        | ( ~select[0] &  select[1] & d[2] )
16        | (  select[0] &  select[1] & d[3] );
17 end
18
19 endmodule
```



Dataflow: VHDL/Verilog

Example: counter in VHDL

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;    -- for the unsigned type

entity COUNTER is
    generic (
        WIDTH : in natural := 32);
    port (
        RST   : in std_logic;
        CLK   : in std_logic;
        LOAD  : in std_logic;
        DATA  : in std_logic_vector(WIDTH-1 downto 0);
        Q     : out std_logic_vector(WIDTH-1 downto 0));
end entity COUNTER;

architecture RTL of COUNTER is
    signal CNT : unsigned(WIDTH-1 downto 0);
begin
    process(RST, CLK) is
    begin
        if RST = '1' then
            CNT <= (others => '0');
        elsif rising_edge(CLK) then
            if LOAD = '1' then
                CNT <= unsigned(DATA); -- type is converted to unsigned
            else
                CNT <= CNT + 1;
            end if;
        end if;
    end process;

    Q <= std_logic_vector(CNT); -- type is converted back to std_logic_vector
end architecture RTL;
```

ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

CSCS Centro Svizzero di Calcolo Scientifico Swiss National Supercomputing Centre

software carpentry

Questions

End of the fourth part