What am I doing here?

Already used in Mateo12
“As below, so above”

• Leverage computer architecture background ...

• ... in higher levels of the system stack

• Looking for further insight
The Programming model osmotic membrane

Applications

PM: High-level, clean, abstract interface

Power to the runtime

What is the right degree of porosity?
Integrate concurrency and data

- Single mechanism
  - Concurrency:
    - Dependences built from data accesses
    - Lookahead: About instantiating work
  - Locality & data management
    - From data accesses

```c
void Cholesky(int NT, float *A[NT][NT]) {
    for (int k=0; k<NT; k++) {
        #pragma omp task inout ([TS][TS](A[k][k]))
        spool (A[k][k], TS);
        for (int i=k+1; i<NT; i++) {
            #pragma omp task in([TS][TS](A[k][k])) inout ([TS])
            vsum (A[k][i], A[k][i], TS);
        }
        for (int i=k+1; i<NT; i++) {
            #pragma omp task in([TS][TS](A[k][i]), [TS][TS])
            inout ([TS][TS](*A[j][i]))
            tsum (A[k][i], A[k][j], A[j][i], TS);
        }
    }
}
```
OmpSs

• A forerunner for OpenMP
Important topics/practices

• Regions
• Nesting
• Taskloops + dependences
• Hints
• Taskify communications: MPI Interoperability
• Malleability
• Homogenize Heterogeneity
• Hierarchical “acceleration”
• Memory management & Locality

Are these topics relevant at ISA level? Can some of these things be leveraged? Are they already there?
Regions

- Precise nD subarray accesses
  - “Complex” analysis but ...

- Enabler for ...
  - Recursion
  - Flexible nesting
  - Taskloop dependences
  - Data management
    - locality
    - layout

```c
void gs (float A[(NB+2)*BS][(NB+2)*BS])
{
    int it,i,j;
    for (it=0; it<NITERS; it++)
        for (i=0; i<N-2; i+=BS)
            for (j=0; j<N-2; j+=BS)
                gs_tile(&A[i][j]);

#pragma omp task
    in(A[0][1;BS], A[BS+1][1;BS], 
      A[1;BS][0], A[1;BS][BS+1]) 
    inout(A[1;BS][1;BS])
void gs_tile (float A[N][N])
{
    for (int i=1; i <= BS; i++)
        for (int j=1; j <= BS; j++)
            A[i][j] = 0.2*(A[i][j] + A[i-1][j] + 
                          A[i+1][j] + A[i][j-1] + 
                          A[i][j+1]);
}
```
Nesting

• Top down
  • Every level contributes

• Flattening dependence graph
  • Increase concurrency
  • Take out runtime overhead from critical path

• Granularity control
  • final clauses, runtime
Nesting

• Top down
  • Every level contributes

• Flattening dependence graph
  • Increase concurrency
  • Take out runtime overhead from critical path

• Granularity control
  • final clauses, runtime
Nesting

• Top down
  • Every level contributes

• Flattening dependence graph
  • Increase concurrency
  • Take out runtime overhead from critical path

• Granularity control
  • final clauses, runtime

J. M. Perez, et all, "Improving the Integration of Task Nesting and Dependencies in OpenMP" IDPS 2017
Taskloops & dependences

- Dependences
  - Intra loop
  - Inter loops

- Dynamic granularities
  - Guided
  - Runtime

- Combination
  - Enabled by regions support
Taskifying MPI calls

• MPI: a “fairly sequential” model

• Taskifying MPI calls
  • Opportunities
    • Overlap/out of order execution
    • Provide laxity for communications
    • Migrate/aggregate load balance issues
  • Risk to introduce deadlocks

• TAMPI
  • Virtualize “communication resource”

V. Marjanovic et al, “Overlapping Communication and Computation by using a Hybrid MPI/SMPSs Approach” ICS 2010
K. Sala et al, "Extending TAMPI to support asynch MPI primitives”. OpenMPCon18
Exploiting malleability

- Malleability
  - Omp_get_thread_num, Thread private, large parallels....

- Dynamic Load Balance & Resource management
  - Intra/inter process/application

- Library (DLB)
  - Runtime interception (MPIP, OMPT, ...)
  - API to hint resource demands
  - Core reallocation policy

- Opportunity to fight Amdalh’s law
  - Productive / Easy !!!
    - Hybridize only imbalanced regions
    - Nx1

“LeWI: A Runtime Balancing Algorithm for Nested Parallelism”, M.Garcia et al. ICPP09
“Hints to improve automatic load balancing with LeWI for hybrid applications” JPDC2014
Homogenizing Heterogeneity

• Performance heterogeneity

• ISA heterogeneity

• Several non coherent address spaces
On the OmpSs road

- **Top down**
  - Lulesh
  - NT-CHEM
  - FFTlib (QE miniapp)
  - Alya

- **Dynamic Load Balance (DLB)**
  - Commutative - multideps
DMRG structure

• Density Matrix Renormalization Group app in condensed matter physics (ORNL)

• Skeleton
  • 3 nested loop
  • Reduction on large array
  • Huge variability of op cost

• Real miniapp
  • Different sizes of Y entries

```
T Y[N];
for (i)
  for (j)
    for (k)
      Y[i] += M[k] op X[j]
```
OpenMP parallelizations

• OpenMP parallelizations
  • Reduction
    • Based on full array privatization
    • Using reduction clauses
  • Nested parallels
    • Worksharings / Tasks
    • Synchronization at end of parallels exposes cost of load imbalances at all levels

• Overheads at fine levels

• Issues activation of multiple levels
  • Core partition
  • Levels of Privatization
Taskification

- Serialize reductions
  - Multiple dependence chains

```c
T Y[N];

for (i)
    for (j)
        for (k)
            Y[i] += M[k] op X[j]
```
Taskification

• Serialize reductions
  • Multiple dependence chains

• Split operations
  • Compute & reduce
  • Persist intermediate result
    • Global array of tmps
    • Used in a circular way
    • Enforce antidependence

• Reduce overhead $\rightarrow$ do not split small operation
  • Compute directly on target operand
  • Avoid task instantiation and dependence overhead
  • Avoid memory allocation, initialization & reduction to target operand
Resulting dependence chains
Performance?

- Causes of pulsation of red tasks?
  - Instantiation order and granularity
  - Graph dependencies

- Improvements
  - Priorities
  - Anti-dependence distances
  - Nesting

Do these effects happen also at ISA level?
Can similar techniques be used to improve performance?
Question on graph scheduling dynamics

$$F = m\ddot{x} - b\dot{x} - kx$$

$$\omega_0 = \sqrt{\frac{k}{m}}$$

$$x(t) = Ae^{-\gamma t} \cos(\omega t - \phi)$$

$$F(t) = F_0 \cos(\omega t) \rightarrow x(t) = A \cos(\omega t)$$

Effective k, m, b ?

Excitation ? Graph generation ?

Resources ?
Thanks to Yale

“There is no limit to what you can achieve provided you do not care who takes the credit”

I first heard it from Yale
Thanks!
Thanks