Toward a Unified Task-based Parallel Programming Interface (UTP)

UPPSALA UNIVERSITET

Afshin Zafari, Elisabeth Larsson

IT Department, Division of Scientific Computing

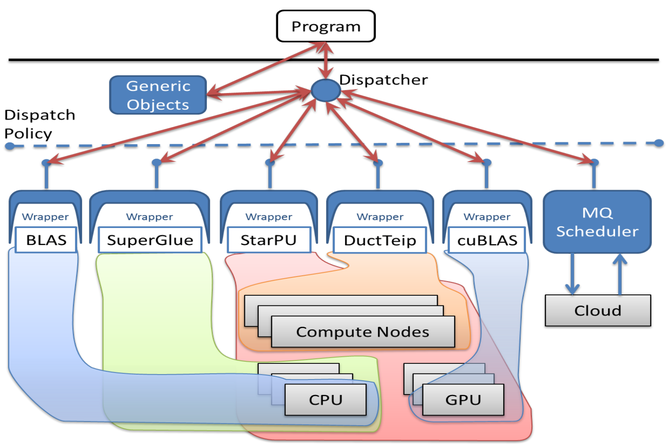
A sample GEMM program

```
int main(int argc, char **argv)
{
  utp_initialize(argc,argv);        // UTP start

  int M = config.getYDimension(); // Get parameters
  int B1 = config.getYBlocks(1);
  int B2 = config.getYBlocks(2);
  GData A(M,M),B(M,M),C(M,M);       // Define Data
  GPartitioner P1(B1,B1);
  GPartitioner P2(B2,B2);           // Define Partitions
  P1->set_next(P2);                 // two levels of partitions
  A.set_partition(P1);              // Apply partitioninng on data
  B.set_partition(P1);
  C.set_partition(P1);
  ugemm(A,B,C);                     // Call unified GEMM on data

  utp_finalize();                   // UTP waits for all tasks
}
```

ugemm

```
void ugemm(GData &A,GData &B,GData &C,GTask *p=NULL){
  int m = A.get_part_countY();
  int n = B.get_part_countX();
  int o = C.get_part_countX();
  for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
      for (int k = 0; k < o; k++)
        ugemm_t(A(i,k),B(k,j),C(i,j),p);
}
void ugemm_t(GData &A,GData &B,GData &C,GTask *p){
  packArgs( args, A  , B  , C     );
  packAxs ( axs , In , In , InOut );
  get_dispatcher()->submit_task(ugemmo,args,axs,p);
}
void ugemmo::split(GTask *t){
  //unpack arguments of t to A,B,C: A = t->args[0] ,...
  ugemm(A,B,C,t);
}
```

Which scheduler or framework?



- **Wrappers** translate framework interfaces to the unified interface.
- **BLAS** and **cuBLAS** wrappers used as schedulers that run kernels.
- **Message Queue (MQ) Scheduler**
  - puts the commands received from Dispatcher to a public queue.
  - gets messages from a public queue and sends them to Dispatcher.

More information?

Search in the *www.it.uu.se* for the keywords **Unified Interface**, **Task Based Parallel Programming**, **Programming Frameworks** or scan this:

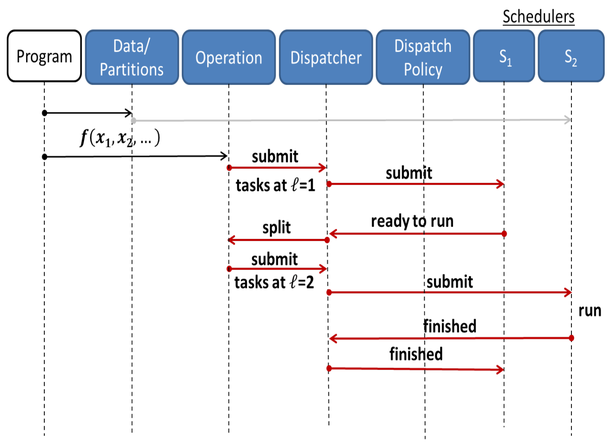
What is it?

- Provides a **unified** programming interface for task based parallel programming.
- Enables **different task-based frameworks** to cooperate without knowing each other.
- Enables a program written ***once*** in **sequential** form to run in **parallel** on different types of computing resources.

Generic Objects

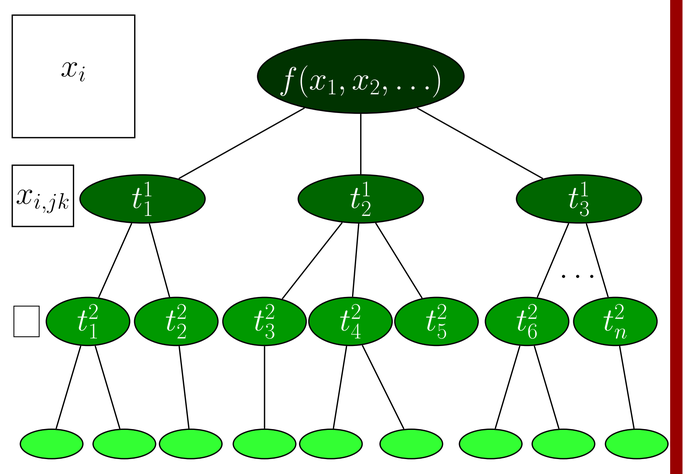
- **Generic Data**
  `M, N, memory, lead_dim, partition, parent, level`
- **Generic Partition**
  `MB, NB, parent, get_part(i,j)`
- **Generic Task**
  `Operation, parent, args[], access[], level, kernel`
- **Generic Operation**
  $f(x,y) \mapsto \{tasks\,(x_{ij}, y_{k\ell})\}$
- **Generic Scheduler**
  `submit(), run(), finished()`
  Notifications (ready and finished)
- **Dispatch Policy**
  Used for customizing the chain of schedulers
- **Dispatcher**
  `submit(), run(), finished()`
  Central hub of *schedulers* conversations.

How does it work?



- Resulting tasks of calling an Operation are submitted to Schedulers via Dispatcher.
- Ready task at any level of the hierarchy splits again using the corresponding Operations.
- Tasks at the finest level run and when finished, Schedulers at higher levels get notified.

Task and Data Hierarchy



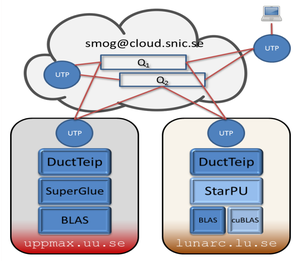
t_m^ℓ: task m at level ℓ
$x_{i,jk}$: partition jk of x_i

Where do the tasks run?

The configuration of computing resources can be determined at run time.

```
run ugemm_app
--cores <multi-core configuration>
--gpus <gpu configuration>
--nodes <dist.  mem.  configuration>
--cloud <cloud configuration>
--cluster <cluster configuration>
```

Configuration Sample



Application programs that use Unified Task Interface can run on cluster, cloud and client computers. Using MQ Scheduler, all instances of UTP programs can communicate tasks and data with each other.

Why to use it?

- *Independent* from frameworks
- *Transparent* to new technologies in the underlying hardware
- *Decoupled* application programming
- *Mixing* different frameworks
- *Single* application program for any available parallelism
- *New* features in any framework become available to *all* application programs (e.g. DLB)
- *Customized* schedulers and dispatch policies can implement new work-flows