A UPC Specification Extension Proposal for Hierarchical Parallelism
Version 1.0

Olivier Serres, Abdullah Kayi, Ahmed Anbar, Tarek El-Ghazawi
GWU - HPCL

{serres, apokayi, anbar}@gwmail.gwu.edu, tarek@gwu.edu

September 23, 2009

This document describes a proposed extension to the UPC language specification which has NOT been ratified by the UPC Consortium. The information presented here does not necessarily reflect the position or the policy of the UPC Consortium and no official endorsement should be inferred.
Contents

1 Rationale
   1.1 Example 1: summation using hierarchical parallelism . . . . 3

2 Specification changes 5

3 Additional notes 6
1 Rationale

The use of multicore processors and hardware accelerators is becoming the norm in modern computing, especially for HPC applications. In a sense this created a processing hierarchy with multiple levels of parallelism. In one possible scenario computational nodes will constitute the top level of parallelism, the cores within a node will be another, and the accelerator processing elements can make the third level. In order for UPC to maintain its ability to efficiently address modern architectural advances, programmers should have the ability to express those changes at the abstract level using a hierarchy of parallelism. One logical place to introduce such hierarchical parallelism in UPC is through nested upc forall. Such use can provide the basis for multilevel workload sharing that matches those new architectural features.

1.1 Example 1: summation using hierarchical parallelism

```c
#include <upc.h>
#include <upc_collective.h>

#define NUM_SUBTHREADS 4

shared int big_array[1024 * THREADS];
shared int sums[NUM_SUBTHREADS * THREADS];
shared int final_sum = 0;

int main(int argc, char **argv){
    upc_forall( int i=0; i<THREADS; i++; i ){
        int *local_ptr = &big_array[MYTHREAD];
        int *local_sums = &sums[MYTHREAD];

        upc_forall( int j=0; j<1024; j++; dynamic(NUM_SUBTHREADS) ){
            local_sums[ upc_mysubthread ] += local_ptr[j];
        }
    }
}
```
In this example, the elements of `big_array` are summed using sub-threads. The `dynamic` affinity in the nested `upc forall` specifies that 4 sub-threads will be spawned for the execution of the loop body.

A dynamic `upc forall` globally behave like a normal `upc forall` with the following differences:

- There is an implicit synchronization at the end of the loop.
The variables declared inside the body of the loop are local and visible only by the sub-thread (like the loop counter \( j \) in this example).

- \texttt{upc\_mysubthread} provides an identifier to the current sub-thread (in this example, it will range from 0 to 3).

2 Specification changes

In this section, the proposed changes to the UPC Language specification v1.2 are addressed.

Section 3 Terms, definitions and symbols

1 sub-thread

an instance of execution initiated by a thread at the \texttt{upc\_forall} statement with dynamic affinity.

Section 5.1.2.3 Program execution

2 Sub-threads cannot perform collective operations.

Section 6 Keywords

3 Add \texttt{upc\_mysubthread} and \texttt{dynamic} to the list of upc keywords.

Section 6.6.2 Iteration Statements

4 Add \texttt{dynamic} as a part of the affinity field. \texttt{dynamic} takes one argument, the number of sub-threads to be created for the loop. Thus, \texttt{dynamic(4)} will create 4 sub-threads to execute the loop iterations. Loop iterations are distributed in a round-robin fashion to the sub-threads.
Semantics

5 When affinity is \texttt{dynamic}, \texttt{upcforall} is no longer a collective operation at that level. The controlling expression and affinity is still single-valued.

6 Private variables can also produce side effects resulting in an undefined behavior.

7 Variable declared inside the body of the loop are not viewed by the other loop iterations and they cannot produce side-effects.

8 When affinity is \texttt{dynamic}, there exists an implicit synchronization after the \texttt{upcforall}.

3 Additional notes

- Due to the memory model, it is important that the loop counter is declared inside the \texttt{upcforall}: the loop counter has to be local to a sub-thread in order for each sub-thread to see a different value. It is recommended for compilers to test for this common mistake and issue a warning if necessary.

- The \texttt{dynamic} keyword may be extended in the future by adding more arguments.