並列分散処理環境における タスク割り当てライブラリの設計と C++での実装と評価

Design, Implementation and Evaluation of Task Mapping Library for Parallel Distributed Processing in C++

山崎　健生　　中山　雅哉
Takeo Yamasaki　　Masaya Nakayama
東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo

1　はじめに

近年の計算機環境はマルチコア・グリッド・クラウドと並列分散化が進んでいる．さらに今後は大型計算機と端末・センサ等が連携したユビキタス・コンピューティングの時代が到来すると考えられる．このように環境の複雑化が予測される中，並列分散処理アプリケーションの開発効率化が必要とされ，多く言語やパラダイムが検討されている．

今回我々は，その中から明示的にタスクを資源に割当てるパラダイムに着目し，新たに C++用ライブラリとして設計した．その設計に際しては C++でのメモリ資源割当てライブラリを参考にしており，メモリ資源と計算機資源を統一的に扱えるような構造となっている．さらにメモリ割当てのように，割当て先やその管理方法に依存せずにタスク構造やアルゴリズムを記述できる．これにより拡張性と移植性の向上を目指す．今回はベースとなる計算機資源の割当て部分を実装し，その性能をベンチマークプログラムにより評価した．

2　ライブラリの構造

このライブラリでは，計算機資源に対してタスクの割り当てをユーザが明示的におこなうモデルをベースとしている．C++では，資源の割当て問題として，メモリ資源の割り当てやデータ構造，データ処理アルゴリズムに関したものが既に標準化されている．今回設計したタスクの割当てライブラリではそのモデルを模倣している．

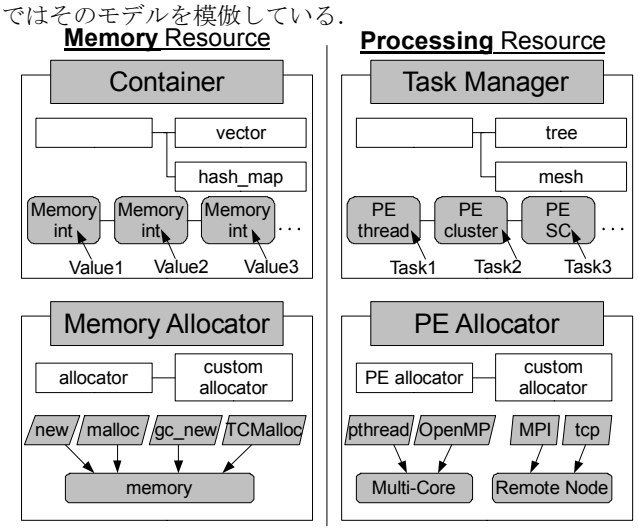

fig. 1　Library structure

fig. 1 において，左側が既存のメモリ割り当て及び，データ構造のライブラリ構造を示す．アロケータによって割り当てたメモリ領域を用い，コンテナが特定のポリシーにしたがって記憶資源を構造化する．そしてコンテナの要素に対し値が代入される形でデータ構造が実現される．それに対して右側がタスク割当てライブラリの構造を示している．PE アロケータによって計算機資源が予約され，タスクマネージャによって特定のポリシーにしたがって計算機構造が仮想的に構築される．その計算機資源に対してタスクが割り当てられ，任意のタスク構造が並列分散的に実行される．

特徴としては拡張性・移植性の高さが挙げられる．C++では，メモリ割り当てにおいて不満があった場合，自作コンテナやカスタムアロケータを作り，その性能を向上させることが可能である．今回設計したタスク割り当てライブラリについても，自作マネージャやカスタムアロケータを作成することができる．そのため，高速な低級ライブラリが新規に登場しても，上位レイヤの変更なしに対応することが出来る．これにより拡張性の向上を狙っている．また，ある程度の制約は必要であるものの，上位レイヤは計算機資源の割当先に依存しないので，環境間の移植性も向上する．

3　プログラミングインターフェース

3.1 Task Mapping

タスク割当てのインターフェースは，C++0x から標準ライブラリに導入される std::thread のインターフェースを参考にしている．

C++0x thread Interface

```
std::thread th( func1, arg1 );
th.join();
```

Task Mapping Interface

```
task_manager   tm();
tm.talloc( func1, arg1, arg2);
tm.talloc( func2, arg3, arg4);
tm.join_all();
```

fig. 2　Task allocation interface

fig. 2 の左側がスレッドのインターフェースで，右側がタスク割り当てのインターフェースとなる．実装では boost ライブラリの thread と同様に C++98 の文法の範囲内で実現されている．このため現在普及している多くのコンパイラで実行可能である．

3.2　Parallel Block

プログラミングインターフェースとして for 文や while 文の並列分散展開や，ブロックの非同期実行をサポートしている．この実装には，ローカルクラスとテンプレート，マクロ関数を利用しており，各ブロックの中はローカル関数となっている．割り当て先にはタスクマネージャを指定する．

3.2　データ転送

サイズの大きいデータの転送用のインターフェースとして remote_memcpy(dst, src, size);といった関数を暫定的に用意している．これをタスクマネージャに渡すと遠隔書き込みが可能となる．

4　まとめ・課題

今回，タスク割当てライブラリを設計し，C++にて一部(PE Allocator)を実装・評価した．ライブラリ構成は既存の C++標準ライブラリのメモリ割当てを模倣しており，インターフェースは C++0x から導入される std::thread を参考にしている．ユーザの手でカスタマイズ可能で，様々低級ライブラリとの共存が可能で拡張性・移植性が高い．C++98 で動作し，既存のデバッガや統合開発環境を用いた開発が可能である．

現在，分散メモリ環境や共有メモリ構造等の複数の環境に対する割当てができるものの，計算機構造やタスク構造を考慮した適応資源割当て機能の実装が課題として残っている．またクラウド環境や，GPU への割り当て等，より多くの環境に対応していく必要があるとともに，分散オブジェクトや分散配列といった生産性を上げる高いレイヤのインターフェース開発が課題となっている．