FUJITSU

shaping tomorrow with you

「富岳」におけるOpenMPのタスク並列

2019年11月5日

富士通株式会社
第三基盤ソフトウェア事業部第一開発部
金子　正教

発表内容

FUJITSU

- OpenMPのタスク機能の紹介
  - 言語仕様
  - サポート範囲
  - 記述例
  - タスクの実装
    - ライブラリを用いた実装
    - libompについて
    - 他社のlibomp対応
  - 富岳の実装
    - コンパイラ
    - ライブラリ
    - C/C++の場合
  - まとめ

OpenMP タスク機能の紹介

FUJITSU

■ task構文の仕様

```
!$omp task  [clause[[,] clause] ...]
  structured-block
!$omp end task
```

指示節 (*clause*) **は以下のいずれか** **赤字はOpenMP4.5仕様** **青字はOpenMP5.0仕様**

```
if ([ task: ] scalar-logical-expression )
final ( scalar-logical-expression )
untied
default ( private | firstprivate | shared | none )
mergeable
private ( list )
firstprivate ( list )
shared ( list )
depend ( dependance-type : list )
priority ( priority-value )
in_reduction ( reduction-identifier : list )
allocatable ( [ allocator :] list )
affinitiy ( [ iterator ( iterators-definition ) :] locator-list )
detach ( event-handle )
```

富岳のサポート範囲 (初期公開時)

FUJITSU

task構文

OpenMP4.0仕様	○
OpenMP4.5仕様	
`priority`指示節	△ (動作保証)
OpenMP5.0仕様	
`in_reduction`指示節	○
`allocatable`指示節	×
`affinity`指示節	×
`detach`指示節	×

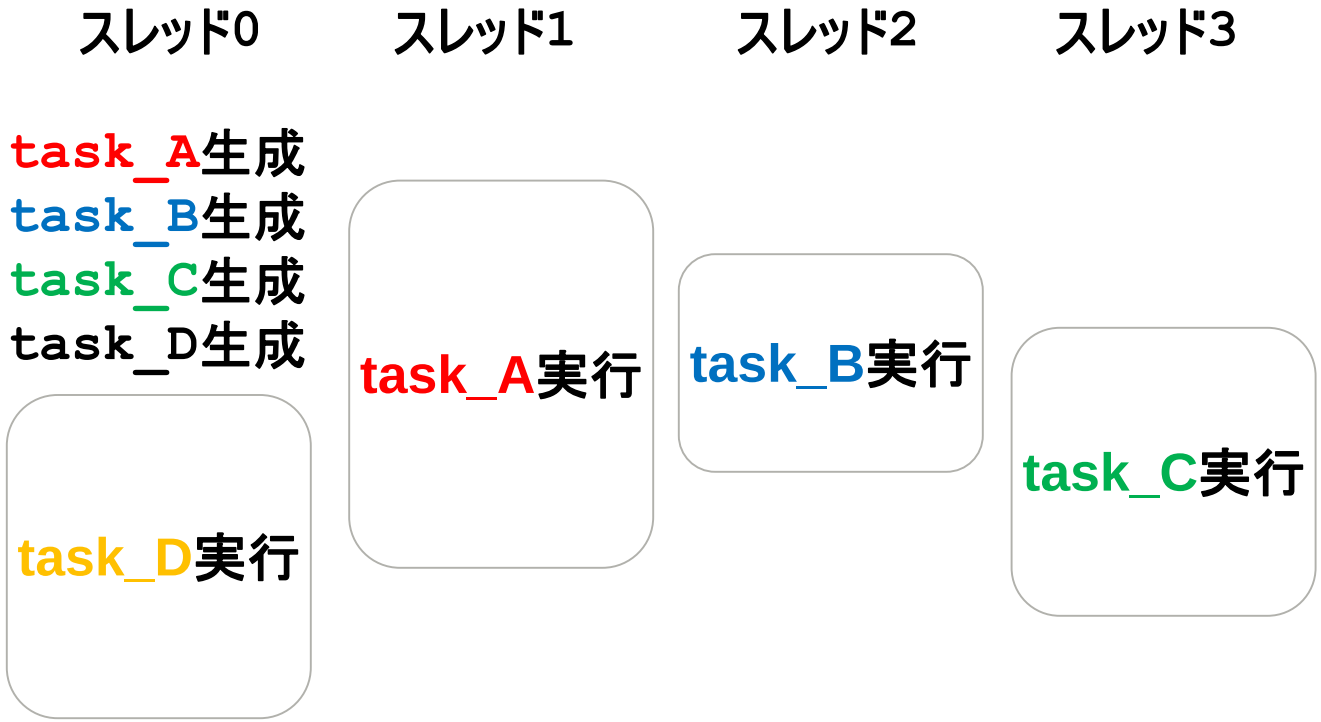
OpenMP タスク機能の紹介

FUJITSU

task構文の記述例

```
!$omp parallel num_threads(4)
!$omp single
!$omp task
      task_A
!$omp end task
!$omp task
      task_B
!$omp end task
!$omp task
      task_C
!$omp end task
!$omp task
      task_D
!$omp end task
!$omp end single
!$omp end parallel
```

動作イメージ



タスク機能の実装

FUJITSU

ライブラリを用いて実装

- ライブラリの機能
  - タスク構造体の生成
  - タスクの起動

ソース

```
      subroutine foo()
!$omp task
        …
!$omp end task
      end subroutine
```

オブジェクト （疑似コード）

```
void _foo()
{
  // タスク構造体の生成
  task = _omp_task_alloc(…);

  // タスクの起動
  _omp_task(task);
}
```

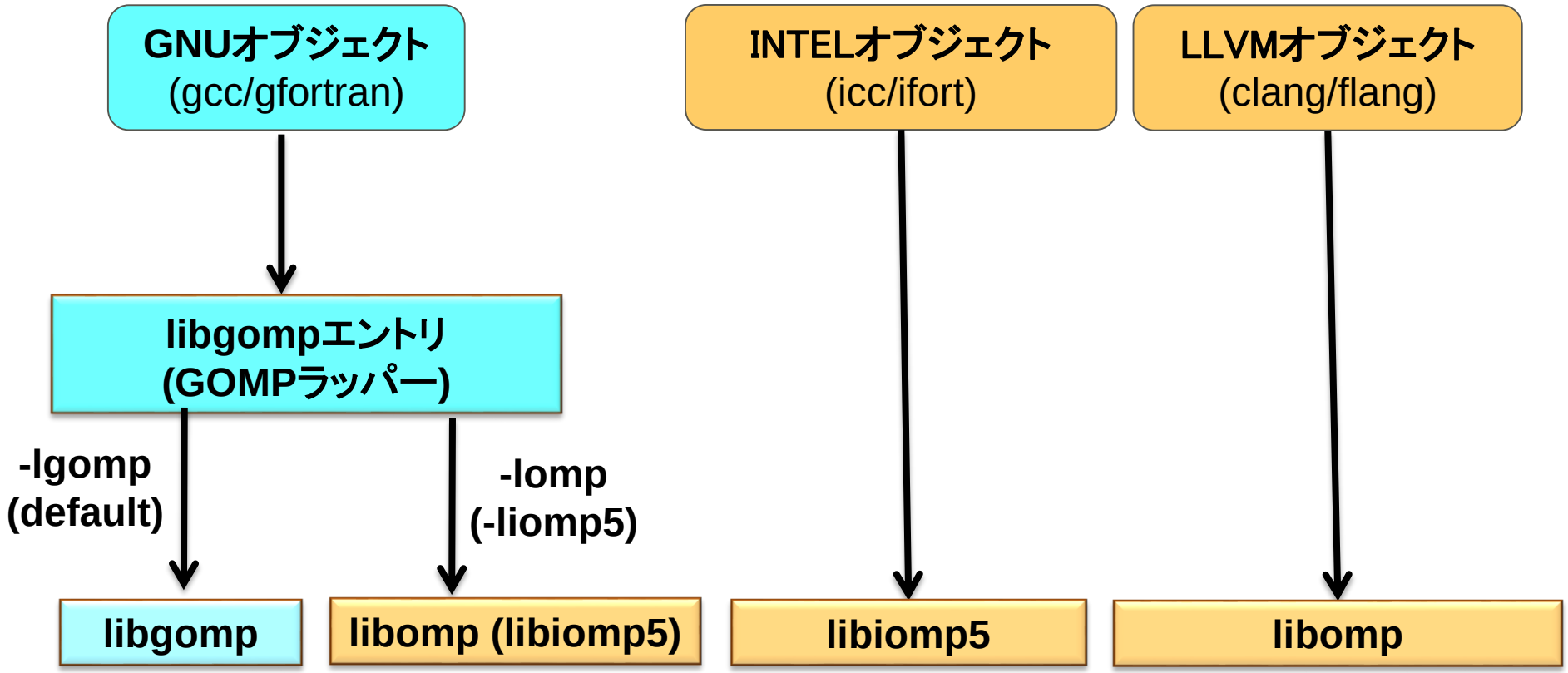
- OpenMPの機能(タスク機能を含む)を実現する libomp というランタイムライブラリがある
- 京ではOpenMP3.1の機能を独自ライブラリで実装
- 富岳のコンパイラでは libomp を採用して、OpenMP4.0以降を実装

libomp

FUJITSU

- libompとは
  - OpenMPのランタイムライブラリ
  - 正式名称：　LLVM OpenMP Runtime Library
  - libompという名称は、ライブラリ名（libomp.so）から
  - IntelのOpenMPライブラリ libiomp5（OSS版）がLLVMプロジェクトに取り込まれたもの
- 業界標準
  - ARM,Intelの各商用コンパイラで採用されている
  - GCCとオブジェクト互換がある
- ライセンス
  - University of Illinois/NCSA Open Source License（LLVMと同じ）と、MIT Licenseのデュアルライセンス
  - IntelのCopyright、および使用許諾契約
  - ARMの使用許諾契約

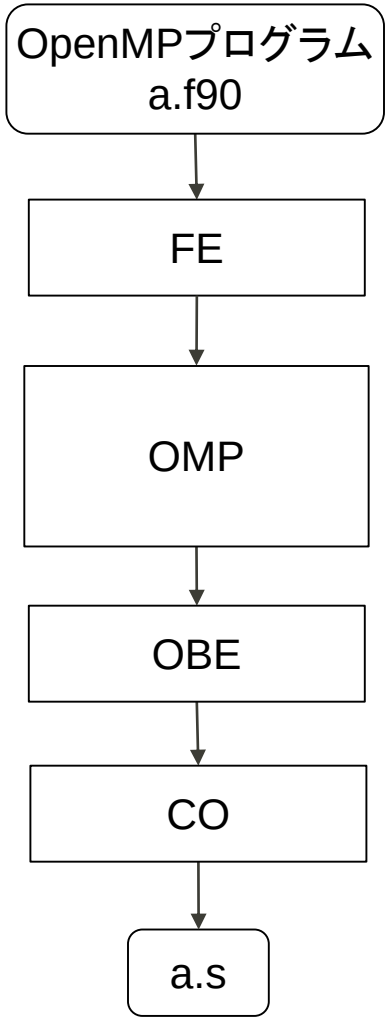
他社の libomp 対応



富岳の実装 (コンパイラ)

FUJITSU

■ 処理フロー



構文解析、コンパイラ内部構造の生成

OpenMP指示行に対する処理
ラッパーを介したOpenMPライブラリ(libomp)の呼出し生成、
コード変換(collapse)、最適化指示(simdなど)の生成

最適化（simd化、自動並列化など）

コード生成

富岳の実装 (ライブラリ)

FUJITSU

- タスクを libompで実装
  - タスク構造体の生成: __kmpc_omp_task_alloc()
  - タスクの起動: __kmpc_omp_task()
  - private変数のメモリ割り付けの考慮
    firstprivate変数の生成とコピーをオブジェクトで行う
- libompの富岳向け独自拡張
  - コア間ハードバリア対応
  - セクタキャッシュ対応

富岳の実装 (C/C++)

FUJITSU

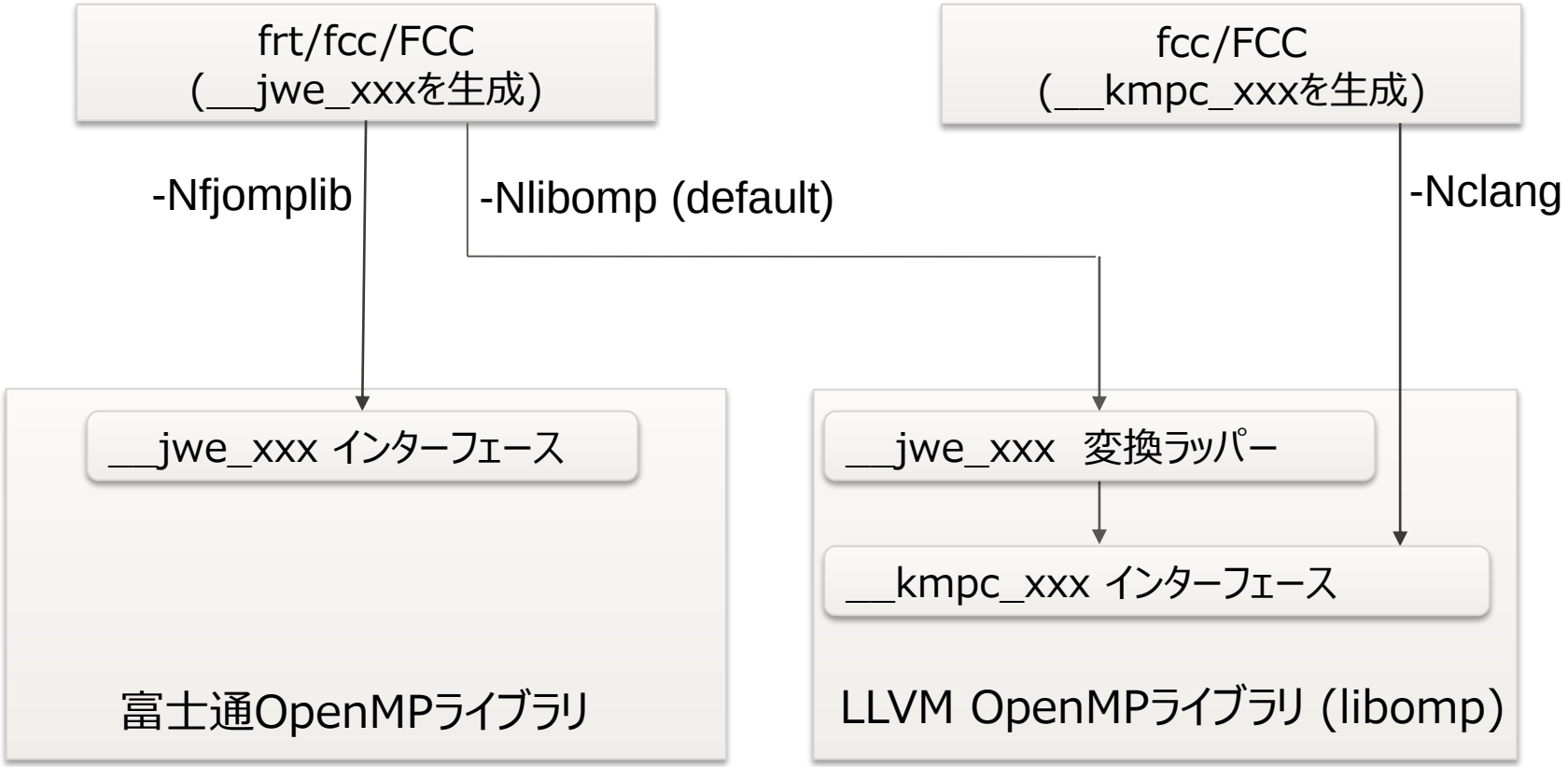
- 【clangモード】 OpenMP4.0以降の仕様をLLVMベースで実装
  -Nclang オプション
  コンパイラ　：　LLVMベース
  ライブラリ　：　LLVM OpenMPライブラリ（libompベースの富岳向け拡張ライブラリ）

- 【tradモード(default)】 京との互換のあるコンパイラとランタイム
  -Nnoclangオプション
  コンパイラ　：　OpenMP3.1+OpenMP4.0のsimd機能
  ライブラリ　：　富士通OpenMPライブラリとLLVM OpenMPライブラリをオプションで切り替え

	clangモード -Nclangオプション	tradモード -Nnoclangオプション
LLVM OpenMPライブラリ(libomp) -Nlibompオプション	結合可	結合可 (default)
富士通OpenMPライブラリ -Nfjomplibオプション	結合不可	結合可

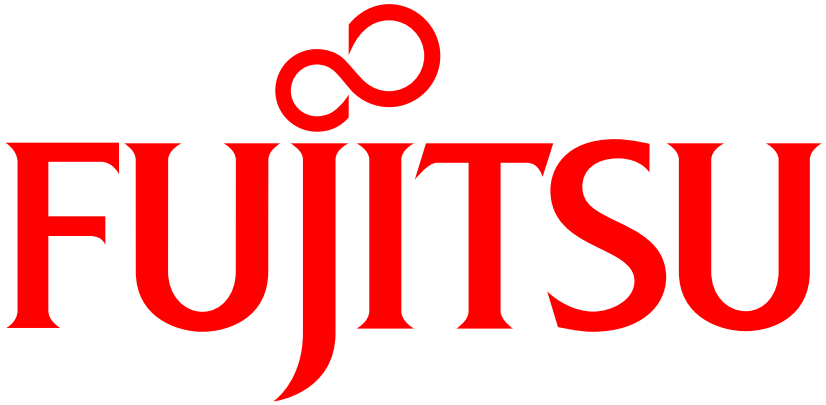
富岳の実装 (ライブラリ)

- LLVM OpenMPライブラリ内に
  富士通OpenMPインタフェースを変換するラッパーを実装



まとめ

- タスク機能の富岳での実装を紹介しました
- 京との互換を持ちつつ、新しい実装にチャレンジしています
- 富岳の完成後もOpenMPの規格に追随します



shaping tomorrow with you