（様式４）

二国間交流事業 共同研究報告書

平成 23 年 4 月 8 日

独立行政法人日本学術振興会理事長　殿

共同研究代表者所属・部局　東京大学・大学院情報理工学系研究科

（ふりがな）　　（いしかわ　ゆたか）
職・氏 名　教授・石川　裕

1. 事　業　名　相手国（　フランス　）との共同研究　　振興会対応機関（　仏外務省　）

2. 研 究 課 題 名　次世代クラスタにおける通信とI/O処理のためのフレームワーク

3. 全 採 用 期 間

平成 21 年 4 月 1 日 ～ 平成 23 年 3 月 31 日 （ 2 年　ヶ月）

4. 研究経費総額

(1) 本事業により交付された研究経費総額　1800　千円

初年度経費 1000 千円、　２年度経費 800 千円、　３年度経費 0 千円

(2) 本事業による経費以外の国内研究経費総額　3171.363　千円

5．研究組織

(1) 日本側参加者

氏名（ふりがな）	所　属・職　名	研　究　協　力　テ　ー　マ
(H21 年度)		
松葉浩也	東京大学情報基盤センター・助教	通信と I/O 処理のためのフレームワーク
野村哲弘	東京大学情報理工学系研究科・博士課程学生	通信と I/O 処理のためのフレームワーク
安達知也	東京大学情報理工学系研究科・修士課程学生	通信と I/O 処理のためのフレームワーク
太田一樹	東京大学情報理工学系研究科・修士課程学生	通信と I/O 処理のためのフレームワーク
(H22 年度)		
美添一樹	東京大学情報理工学系研究科・助教	通信と I/O 処理のためのフレームワーク
野村哲弘	東京大学情報理工学系研究科・博士課程学生	通信と I/O 処理のためのフレームワーク
下沢拓	東京大学情報理工学系研究科・博士課程学生	通信と I/O 処理のためのフレームワーク
路星洋	東京大学情報理工学系研究科・修士課程学生	通信と I/O 処理のためのフレームワーク
Tatiana MARTSINKEVICH	東京大学情報理工学系研究科・修士課程学生	通信と I/O 処理のためのフレームワーク

(2) 相手国側研究代表者

所属・職名・氏名　　ボルドー1 大学　計算機科学科・教授・Namyst Raymond

(3) 相手国参加者（代表者の氏名の前に○印を付すこと）

氏　　名	所属・職名（国名）	研　究　協　力　テ　ー　マ
(H21-22 年度)		
○Namyst Raymond	ボルドー1 大学計算機科学科・教授	通信と I/O 処理のためのフレームワーク
Alexandre DENIS	INRIA,Research Scientist	通信と I/O 処理のためのフレームワーク
Gullaume MERCIER	ボルドー大学・講師	通信と I/O 処理のためのフレームワーク
Francois TRAHAY	ボルドー大学・博士課程学生	通信と I/O 処理のためのフレームワーク
Jerome CLET-ORTEGA	ボルドー大学・博士課程学生	通信と I/O 処理のためのフレームワーク

6. 研究概要（研究の目的・内容・成果等の概要を簡潔に記載してください。）

4〜6 基の CPU コアを搭載した CPU チップが普及し、一台の計算機に 4 基の CPU チップ、すなわち、合計 16〜24 基の CPU コアが搭載された計算機が高速ネットワークに接続されたクラスタが普及している。このようなクラスタはマルチコアクラスタと呼ばれている。マルチコアクラスタ上での並列計算では、一台の計算機上で CPU コア数のプロセスが生成され通信路を共有し、並列計算のための通信とファイル I/O のための通信が発生する。通信路が狭いために性能ボトルネックが生じる。共同研究先とは、マルチコアクラスタにおける性能ボトルネックを解消するために、通信と I/O 処理を融合したフレームワークを共同研究し、大きく 2 つの成果を上げた。

1) 通信と I/O 処理の融合手法

共同研究先で研究開発されてきている NewMadeleine 通信フレームワーク上に I/O 処理のためのレイヤとして BMI と呼ばれる通信機構を移植し、PVFS 並列ファイシステムに対する I/O 処理に NewMadeleine が提供している高性能通信機構を利用できるようにした。さらに、本機構を用いたファイル I/O 処理を効率良く処理するためにキャッシュおよび I/O 処理要求の蓄積、リオーダーリング、一括処理機構を実現し、その有効性を検証した。

2) マルチコアクラスタにおけるポータブル MPI 通信ライブラリ

MPI 通信ライブラリ 3 版で標準化される非同期集団通信を実現するためにはアプリケーション実行スレッドとは別に集団通信を処理するための実行スレッド（これを通信プログレススレッドと呼ぶ）が必要となる。従来の通信プログレススレッド実装手法では pthread を用いて実装されているために効率良く実現できていない。一台の計算機上では一つの通信プログレススレッドをカーネルレベルで実装することにより効率化する手法とユーザレベル軽量スレッド機構を用いた実装を行った。ユーザレベル軽量スレッド実装では、NewMadeleine 通信フレームワークが提供している軽量スレッドを用いた。NewMadeleine 通信ライブラリでは通信とスレッド処理が融合されているために通常の pthread 実装よりも効率良く実現できた。