Vol.2018-HPC-165 No.21
2018/7/31

SX-Aurora TSUBASAにおける プロセス間通信の性能評価

塩月 信智[1,a)]　江川 隆輔[2,b)]　滝沢 寛之[2,c)]

概要：NEC SX-Aurora TSUBASA (SX-AT) は，通常の x86 プロセッサ (Vector Host, VH) のシステムに，PCI-Express を介して独自のベクトルプロセッサ (Vector Engine, VE) を搭載する構成となっている．VH 側では一般的な x86 向け Linux OS が動作しており，VE はその Linux 環境のデバイスの一つとして動作している．VE 用にコンパイルされたプログラムを実行すると，そのプロセスはユーザからは通常のプロセスのように見えるが，物理的には VE 上で動作している．このような機構を採用することで，あたかも標準の Linux 環境のような使い勝手で，VE の持つ高いメモリバンド幅とベクトル演算性能を利用することができる．本研究の目的は，VE 上で動作するプロセス (VE プロセス) と他の通常のプロセス (VH プロセス) との互換性を評価することである．VE プロセスはユーザから VH プロセスと同等に見えるだけではなく，システムソフトウェアからも VH プロセスと同等に見えており，例えばプロセス間通信のような Linux OS が提供する機能もコードの特別な変更なしに利用できる．また，VE プロセスと VH プロセスとの間で通信をすることで，VH と VE という異種プロセッサ間の通信も容易に実現できる．ただし，定量的評価の結果，VH-VE 間のプロセス間通信は通常のプロセス間通信よりもデータ転送性能が低いことが分かった．しかし，NEC が試験実装として現在公開している Vector Engine Offloading (VEO) を使うことで，VH-VE 間の高速なデータ通信を実現できることがわかった．

Performance evaluation of inter-process communication of SX-Aurora TSUBASA

Shinji Shiotsuki[1,a)]　Ryusuke Egawa[2,b)]　Hiroyuki Takizawa[2,c)]

1. 緒言

近年，汎用プロセッサ (Central Processing Unit, CPU) に加えて特徴の異なるプロセッサを混載する複合型計算システムが注目されている．一般的に，Graphics Processing Unit (GPU) や Intel Xeon Phi 等のアクセラレータと呼ばれるプロセッサは，科学技術計算で頻繁に現れるデータ並列性の高い処理の高速実行に適している．異種プロセッサの特長を生かして処理を割り当てることによって，様々な科学技術計算の飛躍的な高速化を達成できることが多数報告されている [1]．

近年 NEC が発表した SX-Aurora TSUBASA (SX-AT) は，通常の x86 プロセッサ (Vector Host, VH) のシステムに，PCI-Express を介して独自のベクトルプロセッサ (Vector Engine, VE) を搭載する構成となっている [2]．VH 側では一般的な x86 向け Linux OS が動作しており，VE はその Linux 環境のデバイスの一つとして動作している．VE 用にコンパイルされたプログラムを実行すると，そのプロセスはユーザからは通常のプロセスのように見えるが，物理的には VE 上で動作している．このような機構を採用することで，あたかも一般的な Linux 環境のような使い勝手で，VE の持つ高いメモリバンド幅とベクトル演算性能を利用することができる．

[1] 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University
[2] 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University
[a)] shinji@sc.cc.tohoku.ac.jp
[b)] egawa@tohoku.ac.jp
[c)] takizawa@tohoku.ac.jp

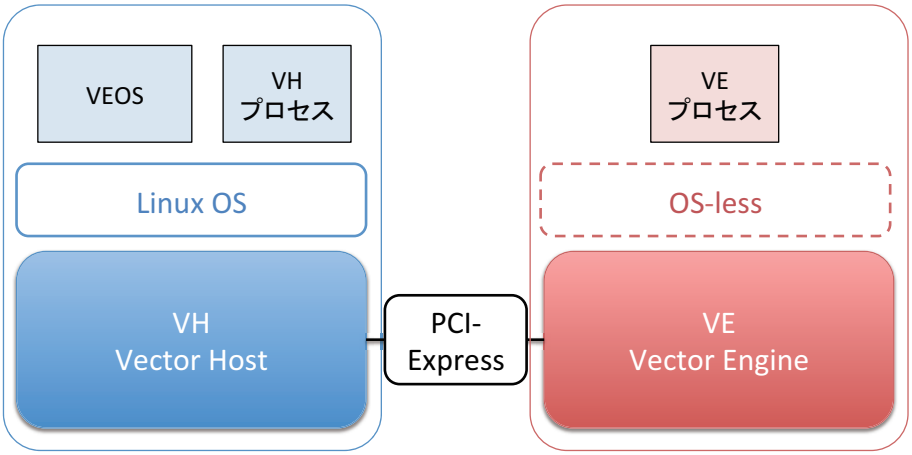

図 1　SX-Aurora TSUBASA のシステム構成

本論文の目的は，VE 上で動作するプロセス (VE プロセス) と，VH 上で動作する通常のプロセス (VH プロセス) との互換性を評価することである．VE プロセスはユーザから VH プロセスと同等に見えるだけではなく，システムソフトウェアからも VH プロセスと同等に見えており，例えばプロセス間通信のような Linux OS が提供する機能もコードの特別な変更なしに利用できることを確認する．また，VE プロセスと VH プロセスとの間で通信をすることで，VH と VE という異種プロセッサ間の通信も容易に実現できることを示す．さらに，プロセス間通信の性能を定量的に評価し，今後の課題を明らかにする．

2. SX-AT のシステム構成

SX-Aurora TSUBASA (SX-AT) は，通常の x86 プロセッサ (Vector Host, VH) のシステムに，PCI-Express を介して独自のベクトルプロセッサ (Vector Engine, VE) を搭載する構成となっている．VE はメモリバンド幅を重視した設計になっており，High Bandwidth Memory (HBM2) と呼ばれるメモリチップ 6 つに並列アクセスすることで最大 1200 GB/s のメモリバンド幅を達成している．図 1 に SX-AT のシステム構成を示す．VH 側では一般的な x86 向け Linux OS が動作しており，VE はその Linux 環境のデバイスの一つとして動作している．VE の制御は，VH 上で動作する VEOS と呼ばれるソフトウェアが担う．VEOS によって，VE のメモリやプロセスの管理，VE 上に立ち上げたプロセスへのプログラムのロード，VE プロセスからのシステムコールの処理等が行われる．このため，ユーザはプログラムを VE 用にコンパイルし実行するだけで，簡単に VE 上でプログラムを実行できる．

VE 用にコンパイルされたプログラムを実行すると，そのプロセスは物理的には VE 上で動作しているが，VEOS の働きによってユーザからは通常のプロセスのように見える．また，VE プロセスはユーザだけでなくシステムソフトウェアからも VH プロセスと同等に見えるため，例えばプロセス間通信のような Linux OS が提供する機能もコードの特別な変更をすることなく利用できる．

このようなシステム構成によって，あたかも標準の Linux 環境のような使い勝手で，VE の持つ高いメモリバンド幅とベクトル演算性能を利用することができる．

3. プロセッサ間通信

本論文では，VE プロセスと VH プロセスとの互換性を議論し，VE プロセスが単にユーザから VH プロセスと同様に見えるだけではなく，システムソフトウェアのレベルでも VH プロセスと同等に見えていることを確認する．VE プロセスが VH プロセスと高い互換性を有していることで，VH 向けに書かれたプログラムが VE 上でも高い確度で動作することが期待できる．その互換性を議論するために，本論文ではプロセス間通信 (Inter-Process Communication, IPC) に着目する．IPC は，複数のプロセス間で情報の共有やメッセージの交換等を行う機能を提供するための仕組みである．基本的に，実行中のプロセスにはプロセスごとに専用のメモリ空間 (仮想アドレス空間) がカーネルによって割り当てられる．これにより，他のプロセスのメモリ空間にはアクセスすることが出来ないため，そのままではデータを共有出来ない．そのため，複数のプロセスで情報を共有するためには IPC を用いる必要がある．そこで，本論文では VH プロセスと VE プロセスとの間の IPC を評価することで，システムソフトウェアのレベルで VH プロセスと VE プロセスが同等に扱えることと，これにより VH と VE との間の通信が実現可能かを確認する．

3.1 プロセス間通信方式の概要

本論文で取り扱う IPC は，名前付きパイプ，POSIX メッセージキュー，Unix ドメインソケット通信，TCP ソケット通信の 4 方式である．これらの IPC 方式は Linux OS が提供する機能である．以下，これらの IPC 方式の概要を簡単に説明する．

名前付きパイプ (以下，First-In First Out，FIFO) は，ファイルのように読み書きが可能なデータ通信経路である [3]．FIFO ファイルをプロセスが読み書きすることでデータのやり取りを行う IPC 方式である．

POSIX メッセージキュー [4](以下メッセージキュー) は，キューと呼ばれるデータ領域を用意し，それを介してメッセージを出し入れすることで通信が可能な IPC 方式である．受信側でのデータの受け取りを確認しないため，この IPC 方式は通信の信頼性を保証しないという欠点があるが，その分だけ高速な通信が実現できるという利点もある．

ソケット通信は，送信側と受信側のプロセスがそれぞれソケットと呼ばれるデータの受け口を用意し，それらを繋ぐことで通信を行う IPC 方式である [5]．ソケットは，マシンの IP アドレスとポート番号の組み合わせによってアドレスが一意に指定される．ソケットを介して送信されたデータは，パケットと呼ばれるチャンクに分割される．これらのパケットは，TCP/IP などのプロトコルによってどのように送信されるか決定される [6]．Unix ドメインソケット通信 (以下 Unix ドメイン) は，同一コンピュータ上での通信を想定したソケット通信であるのに対し，TCP ソケット通信 (以下 TCP) は，インターネットで繋がった外部のコンピュータとデータの送受信が可能なソケット通信である．

3.2 SX-AT におけるプロセッサ間通信

SX-AT は，主に OS 処理を行う x86 プロセッサを搭載する VH とベクトルプロセッサを搭載するベクトルエンジン VE から構成される複合型計算システムの 1 つである．2 節で述べた通り，SX-AT は Linux OS の提供する機能である IPC を使用できる．また，VEOS によって物理的に離れている VE プロセスを VH プロセスと同様に扱うことができる．したがって，SX-AT では，この VH プロセスと VE プロセスの間で IPC を行うことで，VH と VE という異種プロセッサ間の通信が容易に実現可能である．

一般的に，異種プロセッサ上にあるプロセス同士の IPC は，物理的に離れているため時間的コストがかかる．これは，SX-AT の VH プロセスと VE プロセスの間の IPC についても同様である．OS の提供する IPC を利用する場合，プロセスは OS に対してシステムコールを行う必要がある．しかし，SX-AT は VH 上で動作する OS と VE プロセスが離れた場所にあるため，VH プロセスと VE プロセスの間の IPC は，物理的に離れていることに起因する時間的コストに加えて，VE プロセスのシステムコールに伴うオーバーヘッドが生じる．したがって，VE プロセスが OS の提供する IPC の機能を利用する場合，SX-AT のシステム構成が IPC 性能に大きな悪影響を与えることが懸念される．

NEC が試験実装として現在公開している Vector Engine Offloading (VEO) は，VH で動作するプログラムから VE の関数を呼び出す仕組みである [7]．VEO の機能を使うことで，IPC のように VH と VE の間でデータ転送を行うことが可能である．図 2 に VEO のデータ転送のサンプルプログラムを示す．以下，VEO のデータ転送の手順を示す．

(1) VE 側にプロセスを作成する．
(2) プロセスのコンテクストを作成する．
(3) VE 側にメモリを確保する．
(4) コンテクストを使って，VH のメモリ領域を VE のメ

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <ve_offload.h>
5  #define BUFSZ 128
6
7  int main(int argc, char **argv){
8
9      /* 1. create a process on VE */
10     struct veo_proc_handle *proc = veo_proc_create(0);
11     if(proc == NULL) {
12         perror("veo_proc_create");
13         exit(1);
14     }
15
16     /* 2. create a context of the VE context */
17     struct veo_thr_ctxt *ctx = veo_context_open(proc);
18     if(ctx == NULL) {
19         perror("veo_context_open");
20         exit(2);
21     }
22
23     /* 3. prepare for function execution */
24     uint64_t mem;
25     unsigned char* p = (unsigned char*)malloc(BUFSZ);
26     /* allocate a memory region */
27     int ret = veo_alloc_mem(proc, &mem, BUFSZ);
28     if(ret < 0) exit(5);
29     memset(p,0,BUFSZ);
30
31     /* 4. copy VH memory to VE memory */
32     ret = veo_write_mem(proc,mem,p,BUFSZ);
33     if(ret < 0) exit(5);
34
35     /* 5. copy VE memory to VH memory */
36     ret = veo_read_mem(proc,p,mem,BUFSZ);
37     if(ret < 0) exit(5);
38
39     /* 6. close the context */
40     veo_context_close(ctx);
41
42     /* 7. destroy the process */
43     veo_proc_destroy(proc);
44
45     return 0;
46 }
```

図 2　VEO を用いたデータ転送プログラム

モリ領域に複製する．
(5) コンテクストを使って，VE のメモリ領域を VH のメモリ領域に複製する．
(6) コンテクストを終了する．
(7) プロセスを終了する．

VEO の機能を使ってデータ転送を行うことで，VH プロセスと VE プロセス間の高速なデータ転送が可能である．

4. 性能評価と考察

本節では，最初に性能評価環境について述べる．次に，OS の提供する IPC 方式を用いて，SX-AT における IPC の性能評価を行う．最後に，VEO を用いてデータ転送を行う場合の通信性能評価を行う．

4.1 性能評価環境

本節の性能評価は，NEC SX-Aurora TSUBASA A100-1 を用いて行う．表 1 に SX-AT のシステム環境を示す．

表 1 SX-AT のシステム環境

CPU(VH)	Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz
Main Memory	96 GB
OS	CentOS Linux release 7.5.1804
VEOS	veos-1.1.1-1.el7.x86_64
VE Memory Bandwidth	750 GB/s
Compiler for CPU	gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-28)
Compiler for VE	ncc (NCC) 1.2.0 (Build 09:19:58 May 30 2018)

4.2 プロセス間通信の性能

性能評価で使用した IPC 方式は，FIFO，メッセージキュー，Unix ドメイン，TCP の 4 方式である．本実験では，これらの仕組みが異なる 4 つの IPC 方式が SX-AT で正常に動作することを確認しながら，IPC の性能評価を行う．IPC の性能評価は，2 つのプロセスを立ち上げ，それらの間であるサイズのデータを送受信することで行う．本実験の模式図を図 3 に示す．まず，クライアントプログラムのプロセス (プロセス C) がサーバープログラムのプロセス (プロセス S) にデータを送信する．プロセス S は，プロセス C が送信したデータを受け取る．プロセス S が全てのデータを受信したら，今度はプロセス S が同サイズのデータをプロセス C に送信する．プロセス C がプロセス S から送信されたデータを全て受信したら，両プロセスは終了する．最初にプロセス C がプロセス S にデータを送信してから，プロセス S からデータを全て受信するまでの，データの往復にかかる時間を計測し，使用した IPC 方式の read/write の平均バンド幅を求める．2 つのプロセスがそれぞれ VH にある場合と VE にある場合，計 4 通りの場合で行った．2 つのプロセスの配置を表 2 に示す．転送するデータサイズは，128B から 100MB まで変化させて転送速度を測定した．

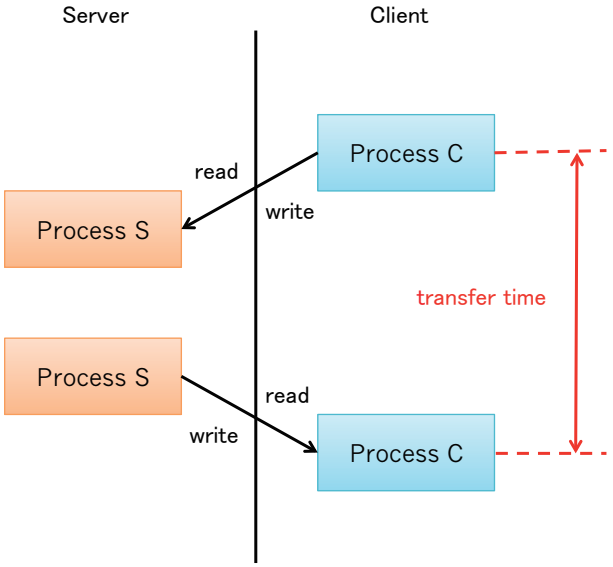

図 3 プロセス間通信の性能評価の模式図

表 2 プロセスの配置による実行パターン

実行パターン	プロセスCの場所	プロセスSの場所
VH-VH	VH	VH
VH-VE	VH	VE
VE-VH	VE	VH
VE-VE	VE	VE

IPC 方式を用いたサーバープログラムおよびクライアントプログラムでは，データの書き込み・読み込みにはシステムコールの read，write を用いた．FIFO は，mkfifo で FIFO ファイルを作成し，データの送受信を行う．メッセージキューは，mq_open でメッセージキューを作成し，データの送受信を行う．ソケット通信である Unix ドメインおよび TCP のプログラムは，ソケットを作成し，送信側と受信側のソケットを接続した後，データの送受信を行う．

プロセス配置の異なる実行パターン VH-VH，VH-VE，VE-VH，VE-VE のそれぞれの場合について，IPC を用いたサーバー・クライアントプログラムを実行した．その結果，どのパターンにおいてもコードの修正をすることなく IPC が正常に動作した．したがって，SX-AT において VH プロセスと VE プロセスの間で両者の区別なく通信できるということがわかった．また，評価を通じて read や write 等のシステムコールに関しても，一般の Linux OS と同様の使い方ができることが確認された．プロセス間通信に限らず，VE プロセスは Linux OS の提供するほぼすべての機能を VH プロセスと同様に使うことができるため [8]，VH 向けに開発された VE でもプログラムが正常に動作する蓋然性が高いと言える．

次に，実行パターン VH-VE，VE-VH の IPC の性能評価結果をそれぞれ図 4，図 5 に示す．図 4 および図 5 において，IPC 方式ごとにグラフの変化をみると，データサイズが大きくなるにつれて転送速度が大きくなっていること

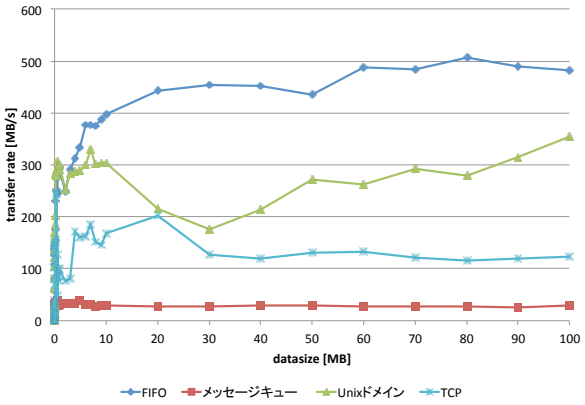

図 4　プロセス間通信の性能評価結果 (VH-VE)

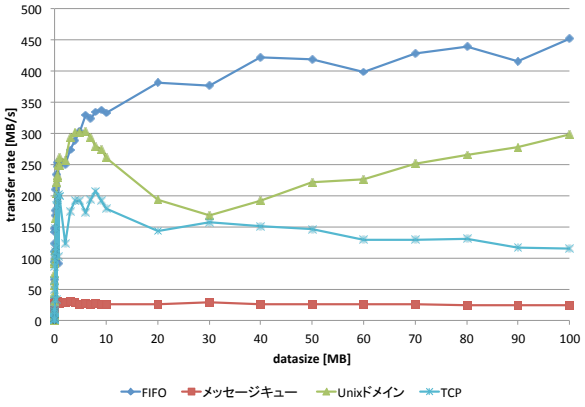

図 5　プロセス間通信の性能評価結果 (VE-VH)

がわかる．これは，データの転送時に必要なメモリアクセスにおいて，一度に多くのデータをやりとりした方が効率的であるためである．また，データサイズが大きくなると転送速度がある値に収束している．これは，メモリアクセス速度がメモリバンド幅に制限されており，データサイズが大きくなるとメモリアクセスがデータ転送における律速となるためである．

IPC は VH-VE 間の通信を容易に実現できる一方で，得られた VH-VE 間の平均バンド幅は最高でも 500 MB/s 程度であった．SX-AT における IPC は，VE プロセスの Linux 環境での正常動作を保証するために利用可能ではあるが，VH-VE 間の高速通信用に用意されているものではない．SX-AT における IPC は，高速通信用ではないことに加え，物理的に離れているというシステム構成の影響があるため，現状ではデータ転送性能が高くないことがわかった．

HPC 分野で高速通信用に用いられる MPI 通信の場合，VE と InfiniBand Host Channel Adapter (HCA) が Linux OS を経由せずに直接データのやり取りを行う (Direct Memory Access, DMA) ことで，VH と VE が物理的に離れている影響を最小限に抑えている．SX-AT でも MPI 通信による高速データ転送は可能であるが，ユーザに MPI プログラミングを強いることになる．SX-AT では，ユーザのプログラムで高速な VH-VE 間の通信を実現するために，VEO という仕組みがある．次節で，VEO のデータ転送性能を評価する．

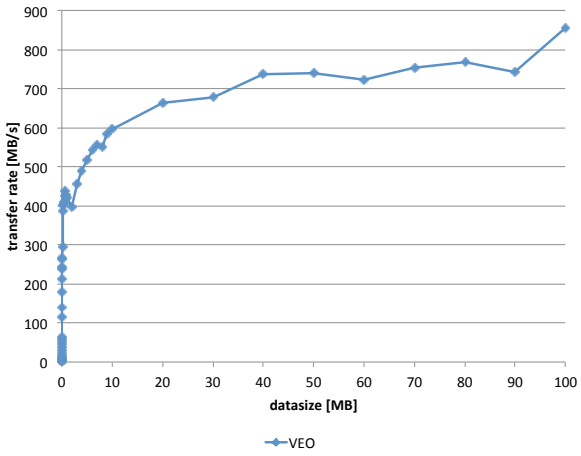

図 6　VEO のデータ転送性能評価結果

4.3　VEO のデータ転送性能

SX-AT における VH-VE 間のデータ転送は，OS の提供する IPC だけではなく，NEC が試験実装として現在公開している VEO の機能によって実現することが可能である．VEO によるデータ転送は，VH のメモリ領域を VE のメモリ領域に複製し，その後 VE のメモリ領域を VH のメモリ領域に複製するというメモリの read/write によって行われる．本実験では，VEO の機能を用いて VH から VE，VE から VH のデータ転送を行い，read/write の平均バンド幅を測定する．

VEO のデータ転送性能評価の結果を図 6 に示す．図 6 から，VEO のデータ転送性能は IPC に比べて非常に高いことがわかった．得られた VEO のデータ転送性能は，GPU における pageable なメモリ領域のデータ転送とさほど変わらない性能を有しており [9]，十分な性能が出ていると考えられる．VH-VE 間の高速な通信が必要な場合には，VEO を使う必要があることが明らかになった．しかし，現状の VEO を使っても PCI-Express の転送能力を使い切ることはできていない．このため，さらなる機能強化や性能改善が期待される．

5.　結言

本論文では，SX-AT の VH および VE でプロセスを実行し，VE プロセスが VH プロセスと同様に Linux のプロセス間通信に関する主要機能を利用できることを明らかにした．VH プロセスと VE プロセスとの区別がないことから，両者を通信させることによって VH と VE との通信を容易に実現することができる．ただし，VE プロセスの Linux 環境での正常動作を保証するためにプロセス間通信機能が利用可能になっているだけで，もともと VH-VE 間の高速

通信用に用意されているものではない．このため，Linuxが提供するプロセス間通信の機能を使ってVHプロセスとVEプロセスを通信させた場合には，低い性能しか達成できないことも示された．そのようなVH-VE間の高速な通信が必要な場合には，NECが試験実装として現在公開しているVEOを使う必要があることも示された．

謝辞

本研究を進めるにあたり，有益なご助言をいただいた日本電気株式会社の今井照之氏，山田洋平氏および政岡靖久氏に感謝します．

本研究の一部は，DFG/JST SPPEXA ExaFSAおよび科研費基盤研究 (B)16H02822の支援を受けている．

参考文献

[1] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E. and Purcell, T. J.: A survey of general-purpose computation on graphics hardware, *Computer graphics forum*, Vol. 26, No. 1, Wiley Online Library, pp. 80–113 (2007).

[2] NEC Corporation: SX-Aurora TSUBASA Program Execution Quick Guide (Japanese) (2018).

[3] J.A. Williams, N. B. and Xie, X.: FIFO communication models in operating systems for reconfigurable computing, Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (2005).

[4] Maxim Ya. Afanasev, Yuri V. Fedosov, A. A. K. and Shorokhov, S. A.: Performance evaluation of the message queue protocols to transfer binary JSON in a distributed CNC system, IEEE 15th International Conference on Industrial Informatics (2017).

[5] Venkataraman, A. and Jagadeesha, K. K.: Evaluation of Inter-Process Communication Mechanisms, URL:http://pages.cs.wisc.edu/∼adityav/projects.html (2015).

[6] Postel, J.: Transmission control protocol, Defense Advanced Research Projects Agency Information Processing Techniques Office 1400 Wilson Boulevard (1981).

[7] NEC Corporation: SX-Aurora Vector Engine Offloading (VEO), URL:https://github.com/SX-Aurora/veoffload (2018).

[8] NEC Corporation: Difference Point foe system calls version 2.3, URL:https://jpn.nec.com/hpc/aurora/vesoftware/index.html (2018).

[9] Hahnfeld J., Terboven C., P. J. P. H. and M.S., M.: Evaluation of Asynchronous Offloading Capabilities of Accelerator Programming Models for Multiple Devices, *Lecture Notes in Computer Science*, Vol. 10732, Springer, Cham (2018).