


2021-12-31

A decorative graphic consisting of several horizontal lines of varying lengths and colors (dark blue, light blue, and grey) stacked on top of each other.

PIM-API Manual

Embedded Systems and Computer Architecture Lab., Yonsei University

Hyoseong Choi <hyoseong.choi@yonsei.ac.kr>
Chanyoung Yoo <chanyoung.yoo@yonsei.ac.kr>
Hongju.kal <hongju.kal@yonsei.ac.kr>
Jeonghoon Choi <jeonghoon.choi@yonsei.ac.kr>
Enhyeok Jang <e@yonsei.ac.kr>

PIM-API Manual

1. PIM Software Overview

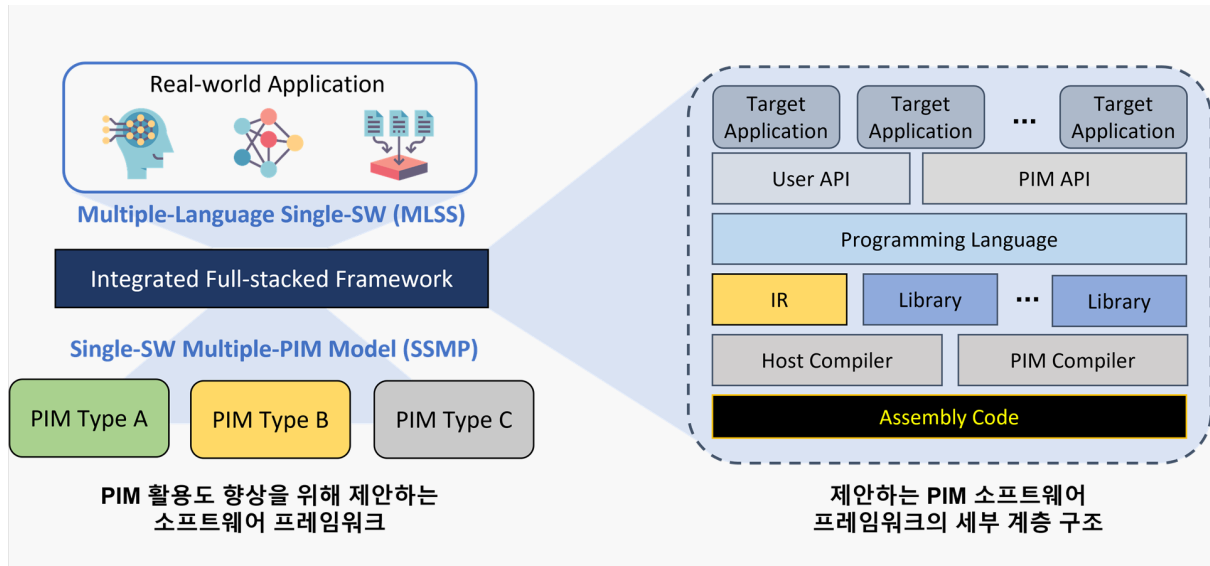


Figure 1. 통합형 PIM 소프트웨어 프레임워크

최근 데이터센터 환경에서 프로세서와 메모리 간 통신에 의한 느린 지연시간 및 큰 전력 소모를 개선하기 위해 Processing-in-Memory (PIM)의 하드웨어 관련 연구 개발이 활발히 이루어지고 있음. 반면, PIM 지원 프로그래밍 언어 및 컴파일러 개발과 같은 소프트웨어 관련 연구 및 개발은 매우 부진한 상황임. PIM 프로그래밍 환경을 구성하는 데 있어, 다종의 PIM 하드웨어로 인해 각기 다른 소프트웨어 스택이 필요하고 통합 환경을 지원하는 데 어려움이 있음. 따라서, 본 연구과제는 통합형 PIM 소프트웨어는 다양한 하드웨어 구조를 고려한 다수의 프로그래밍 언어를 제안함 (MLSS: Multiple-Language Single-Software). 또한, 다수의 PIM 하드웨어 모델을 그룹화하고 통합 컴파일러 환경을 제공하고자 함 (SSMP: Single-Software Multiple-PIM). 추가적으로, 통합형 PIM 소프트웨어를 검증하는데 있어서 필요한 다종의 PIM 시뮬레이터 및 PIM 향 어플리케이션을 제공함. 제안하는 통합형 PIM 소프트웨어 프레임워크는 Figure 1 과 같다.

본 매뉴얼은 PIM API 의 사용 방법에 대해 서술한다.

2. 시뮬레이터 환경 설정 및 구동

A. 개발 및 빌드 환경

Software	Version
Operating System	Ubuntu 18.04.6 LTS (Bionic Beaver)
GCC	7.5.0
clang	6.0.0

B. 설치 및 환경 설정

사전 설정

lib 디렉토리가 없다면, lib 디렉토리를 생성해줘야 한다.

소스코드 컴파일

```
root@cbe6533abfd8:/home/work/PIM-Software/PIM_BLAS# ls
Makefile  bench  include  lib  obj  src
```

Figure 2. PIM_BLAS 디렉토리

1. Build the PIM_BLAS

```
$ make
```

```
root@cbe6533abfd8:/home/work/PIM-Software/PIM_BLAS# make
make[1]: Entering directory '/home/work/PIM-Software/PIM_BLAS/src'
clang -fPIC -c -I../include -o ../obj/cblas_sgemv.o ../src/cblas_sgemv.c
clang -fPIC -c -I../include -o ../obj/cblas_sdot.o ../src/cblas_sdot.c
clang -fPIC -c -I../include -o ../obj/cblas_sscal.o ../src/cblas_sscal.c
clang -fPIC -c -I../include -o ../obj/pim_avail_op.o ../src/pim_avail_op.c
clang -fPIC -c -I../include -o ../obj/cblas_sdsdot.o ../src/cblas_sdsdot.c
clang -fPIC -c -I../include -o ../obj/cblas_sgemm.o ../src/cblas_sgemm.c
clang -fPIC -c -I../include -o ../obj/cblas_saxpy.o ../src/cblas_saxpy.c
clang -shared -Wl,-soname,libpimblas.so -o ../lib/libpimblas.so.0.0.0 ../obj/cblas_sgemv.o ../obj/cblas_sdot.o ../obj/cblas_sscal.o ../obj/pim_avail_op.o ../obj/cblas_sdsdot.o ../obj/cblas_sgemm.o ../obj/cblas_saxpy.o
ln -s ../lib/libpimblas.so.0.0.0 ../lib/libpimblas.so
make[1]: Leaving directory '/home/work/PIM-Software/PIM_BLAS/src'
make[1]: Entering directory '/home/work/PIM-Software/PIM_BLAS/bench'
clang -std=c11 -g -Wall -fopenmp -o ../test test.c -I../include -L../lib -lpimblas
make[1]: Leaving directory '/home/work/PIM-Software/PIM_BLAS/bench'
```

Figure 3. Successful Building

```
root@cbe6533abfd8:/home/work/PIM-Software/PIM_BLAS# ls lib/  
libpimblas.so  libpimblas.so.0.0.0
```

Figure 4. lib 디렉토리

lib 디렉토리에 libpimblas.so, libpimblas.so.0.0.0 가 생겨야 한다.

libpimblas.so, libpimblas.so.0.0.0 파일은 공유 라이브러리로, application 의 make file 을 만들 때, -Lpimblas 을 입력해서 compile 할 수 있도록 한다.

2. *Export library command inside the application directory*

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:(your directory)/PIM_BLAS/lib
```

위의 command 를 application 을 돌리는 디렉토리에서 입력해서 libpimblas.so, libpimblas.so.0.0.0 와 같은 공유 라이브러리를 application 에서 활용할 수 있도록 만든다.

3. PIM-API 기본 벤치마크

Testbench

해당 벤치마크는 cblas 의 각종 API 를 PIM 에 맞춰서 테스트해볼 수 있는 프로그램.

```
$ make
```

PIM_BLAS 디렉토리에서 make command 를 입력하면 자동으로 PIM_BLAS 디렉토리 안에 test 실행 파일이 생긴다.

```
$ ./test
```

./test 를 실행시켜 주면, PIM 에 맞춘 cblas API 들의 결과값을 확인할 수 있다. 나아가, openmp 도 구현하면서 cblas API 를 c 언어에서 뿐만 아니라 openmp 로도 구현할 수 있는 것을 확인할 수 있다.

<pre> result of cblas_sgemv 13204.500000 1.000000 12934.299805 3.000000 13376.600586 5.000000 13967.100586 7.000000 13092.700195 9.000000 12965.000000 11.000000 13549.799805 13.000000 13724.199219 15.000000 13009.400391 17.000000 13024.199219 19.000000 13751.500000 21.000000 13515.500977 23.000000 12954.600586 25.000000 13111.900391 27.000000 13981.700195 29.000000 13341.000000 31.000000 12928.300781 33.000000 13228.100586 35.000000 14240.400391 37.000000 13200.700195 39.000000 </pre>	<pre> result of openmp cblas_sgemv 13204.500000 1.000000 12934.299805 3.000000 13376.600586 5.000000 13967.100586 7.000000 13092.700195 9.000000 12965.000000 11.000000 13549.799805 13.000000 13724.199219 15.000000 13009.400391 17.000000 13024.199219 19.000000 13751.500000 21.000000 13515.500977 23.000000 12954.600586 25.000000 13111.900391 27.000000 13981.700195 29.000000 13341.000000 31.000000 12928.300781 33.000000 13228.100586 35.000000 14240.400391 37.000000 13200.700195 39.000000 </pre>
---	--

Figure 5. Result of Testbench Example (cblas_sgemv)

```

result of openmp cblas_sgemm
670.400024 695.600037 685.099976 669.500000 704.899963 704.600037 697.500061 710.799988 688.400024 683.000000
706.500000 694.300049 699.099976 700.500000 666.200012 681.200012 692.800049 668.700012 693.900024 683.399963
701.000000 677.700012 715.599976 724.600037 711.500000 727.299988 712.500061 709.599976 735.600037 714.000000
721.299988 725.200012 681.500000 699.000000 713.099976 679.600037 707.300049 699.299988 676.000000 713.899963
733.500000 726.300049 715.700012 749.300049 720.000000 719.600037 761.700012 742.599976 752.400024 746.899963
705.700012 725.700012 730.400024 699.400024 729.599976 712.200012 691.400024 731.799988 724.600037 714.000000
720.400024 727.400024 698.700012 722.700012 739.899963 721.400024 753.900024 749.000000 708.400024 749.399963
754.700012 724.300049 743.200012 726.400024 706.200012 735.299988 728.700012 718.700012 725.700012 697.000000
663.600037 679.100037 691.200012 667.600037 693.299988 683.299988 668.200012 704.099976 704.300049 689.200012
703.000000 681.100037 676.200012 700.200012 688.500000 693.799988 695.700012 661.900024 677.400024 689.500000
707.500000 674.500061 702.700012 703.700012 680.899963 719.299988 728.800049 716.200012 732.500000 701.200012
698.799988 725.300049 704.200012 712.000000 716.399963 673.200012 691.200012 705.799988 672.800049 701.000000
721.000000 704.100037 683.799988 741.700012 735.000000 724.900024 759.000061 730.200012 730.300049 747.399963
728.799988 739.100037 734.099976 693.400024 713.899963 719.100037 688.600037 719.299988 702.400024 682.099976
679.400024 707.100037 699.099976 713.200012 718.799988 688.700012 711.300049 727.099976 707.200012 736.599976
730.299988 688.300049 727.900024 731.800049 700.000000 717.500000 699.300049 677.700012 705.400024 697.399963
685.299988 691.100037 693.500000 661.900024 677.899963 690.500000 667.400024 693.599976 684.100037 659.299988
695.700012 696.400024 681.799988 696.100037 674.700012 670.299988 694.800049 683.599976 689.400024 691.799988
710.200012 667.500061 686.000000 711.300049 678.799988 707.500000 709.000061 686.700012 725.600037 716.899963
704.799988 721.600037 690.799988 688.900024 715.899963 695.299988 703.600037 708.500000 665.800049 684.299988
704.700012 710.400024 680.400024 730.300049 713.899963 694.100037 752.500061 746.299988 736.700012 744.099976
715.799988 716.400024 734.000000 715.900024 726.700012 722.200012 682.000061 703.000000 708.700012 708.700012
672.600037 688.700012 669.099976 673.300049 699.599976 690.200012 702.900024 707.099976 675.600037 693.399963
707.799988 686.500061 714.500000 706.800049 663.399963 701.600037 704.100037 670.900024 687.000000 667.399963
670.900024 667.000061 692.000000 684.700012 691.000000 693.900024 662.800049 679.299988 692.400024 657.899963
684.600037 675.600037 651.299988 688.200012 689.399963 675.299988 690.100037 669.200012 665.300049 690.299988
709.100037 689.000061 697.799988 715.100037 672.899963 691.900024 717.700012 685.700012 714.900024 696.500000
674.700012 714.100037 705.900024 694.300049 711.599976 681.299988 679.900024 707.400024 687.300049 696.099976
716.900024 712.900024 673.200012 715.100037 721.299988 691.799988 742.200012 726.299988 707.000000 737.000000
731.299988 722.200012 730.099976 702.300049 703.399963 721.500000 703.900024 715.200012 711.200012 671.500000
635.400024 672.200012 673.299988 667.600037 682.299988 661.299988 664.100037 689.000000 678.200012 684.399963
687.200012 654.300049 670.700012 683.700012 661.000000 687.600037 678.500061 633.700012 670.500000 671.599976
605.000000 671.400024 686.700012 671.400024 668.000000 693.500000 686.700012 693.500000 696.900024 652.700012
669.700012 683.300049 649.299988 676.500000 668.000000 644.200012 681.600037 683.299988 669.700012 685.000000
704.200012 674.400024 673.500000 715.100037 695.500000 704.799988 722.600037 680.900024 700.400024 704.599976
673.100037 702.800049 684.900024 663.600037 703.500000 695.799988 684.700012 702.500000 672.700012 671.799988
693.000000 711.600037 694.500000 728.400024 724.899963 685.700012 728.100037 734.799988 705.800049 726.099976
710.700012 691.900024 722.400024 717.200012 708.599976 717.000000 689.700012 691.299988 709.900024 692.799988
632.400024 657.600037 647.099976 631.500000 666.899963 666.600037 659.500061 672.799988 650.400024 645.000000
668.500000 656.300049 661.099976 662.500000 628.200012 643.200012 654.800049 630.700012 655.900024 645.399963

```

Figure 6. Result of Testbench Example (cblas_sgemm)

4. PIM-API

선정된 BLAS 연산 6 종을 PIM HW 에서 수행하기 위해, 입력으로 들어온 vector 는 16x1 크기로, matrix 는 16x16 크기로 나눈 뒤, 각각의 subarray 에 대해 bank level 에서 병렬 연산을 수행할 수 있도록 각각의 함수를 정의함.

cblas_sdot

1. 개요

BLAS Level 1 연산인 SDOT 연산을 구현.

2. 설명

두 float 벡터 간 내적을 연산한다.

$X^T \cdot Y = \sum_{i=1}^n X(i) \times Y(i)$ 의 값을 구한다.

3. 파라미터

N	int	각 벡터의 element 수
X	float	첫 번째 벡터 피연산자
incX	int	X 의 element 간의 인덱스 증가 값
Y	float	두 번째 벡터 피연산자
incY	int	Y 의 element 간의 인덱스 증가 값

cblas_sdsdot

1. 개요

BLAS Level 1 연산인 SDSDOT 연산을 구현.

2. 설명

두 float 벡터 간 배정밀도 내적을 연산한다.

$\alpha + X^T \cdot Y = \alpha + \sum_{i=1}^n X(i) \times Y(i)$ 의 값을 구한다.

3. 파라미터

N	int	각 벡터의 element 수
alpha	float	내적 결과에 더해지는 스칼라 값
X	float	첫 번째 벡터 피연산자
incX	int	X 의 element 간의 인덱스 증가 값
Y	float	두 번째 벡터 피연산자
incY	int	Y 의 element 간의 인덱스 증가 값

cblas_sscal

1. 개요

BLAS Level 1 연산인 SSCAL 연산을 구현.

2. 설명

float 벡터를 float 스칼라 값만큼 스케일링한다.

αX 의 값을 구한다.

3. 파라미터

N	int	각 벡터의 element 수
alpha	float	스케일링할 스칼라 값
X	float	벡터 피연산자
incX	int	X 의 element 간의 인덱스 증가 값

cblas_saxpy

1. 개요

BLAS Level 1 연산인 SAXPY 연산을 구현.

2. 설명

다른 float 벡터에 float 벡터의 스칼라 배수를 더한다.

$Y \leftarrow \alpha X + Y$ 의 값을 구한다.

3. 파라미터

N	int	각 벡터의 element 수
alpha	float	스케일링할 스칼라 값
X	float	스칼라 곱 피연산자 벡터
incX	int	X 의 element 간의 인덱스 증가 값
Y	float	X 의 스칼라 곱 벡터가 더해질 벡터
incY	int	Y 의 element 간의 인덱스 증가 값

cblas_sgemv

1. 개요

BLAS Level 2 연산인 SGEMV 연산을 구현.

2. 설명

행렬-벡터 간 연산을 수행한다.

$Y \leftarrow \alpha(A \times X) + \beta Y$ 또는 $Y \leftarrow \alpha(A^T \times X) + \beta Y$ 의 값을 구한다.

3. 파라미터

Order		row major/column major 여부를 결정
TransA		Transpose 여부를 결정
M	int	행렬 A의 행의 개수
N	int	행렬 A의 열의 개수
alpha	float	스케일링할 스칼라 값 α
A	float	피연산자 실수 행렬 A
lda	int	행렬 A의 첫 번째 dimension
X	float	피연산자 실수 행렬 X
incX	int	X의 element 간의 인덱스 증가 값
beta	float	스케일링할 스칼라 값 β
Y	float	피연산자 실수 행렬 Y
incY	int	Y의 element 간의 인덱스 증가 값

cblas_sgemm

1. 개요

BLAS Level 3 연산인 SGEMM 연산을 구현.

2. 설명

벡터-벡터 간 연산을 수행한다.

$op(X) = X$ 또는 $op(X) = X^T$ 일 때, $C \leftarrow \alpha(op(A) \times op(B)) + \beta C$ 의 값을 구한다.

3. 파라미터

Order		row major/column major 여부를 결정
TransA		행렬 A의 Transpose 여부를 결정
TransB		행렬 B의 Transpose 여부를 결정
M	int	행렬 A의 행의 개수
N	int	행렬 B의 열의 개수
K	int	행렬 A의 열의 개수 또는 행렬 B의 행의 개수
alpha	float	스케일링할 스칼라 값 α
A	float	피연산자 실수 행렬 A
lda	int	행렬 A의 첫 번째 dimension
B	float	피연산자 실수 행렬 B
ldb	int	행렬 B의 첫 번째 dimension

beta	float	스케일링할 스칼라 값 β
C	float	피연산자 실수 행렬 C
ldc	int	행렬 C 의 첫 번째 dimension