AliCloud

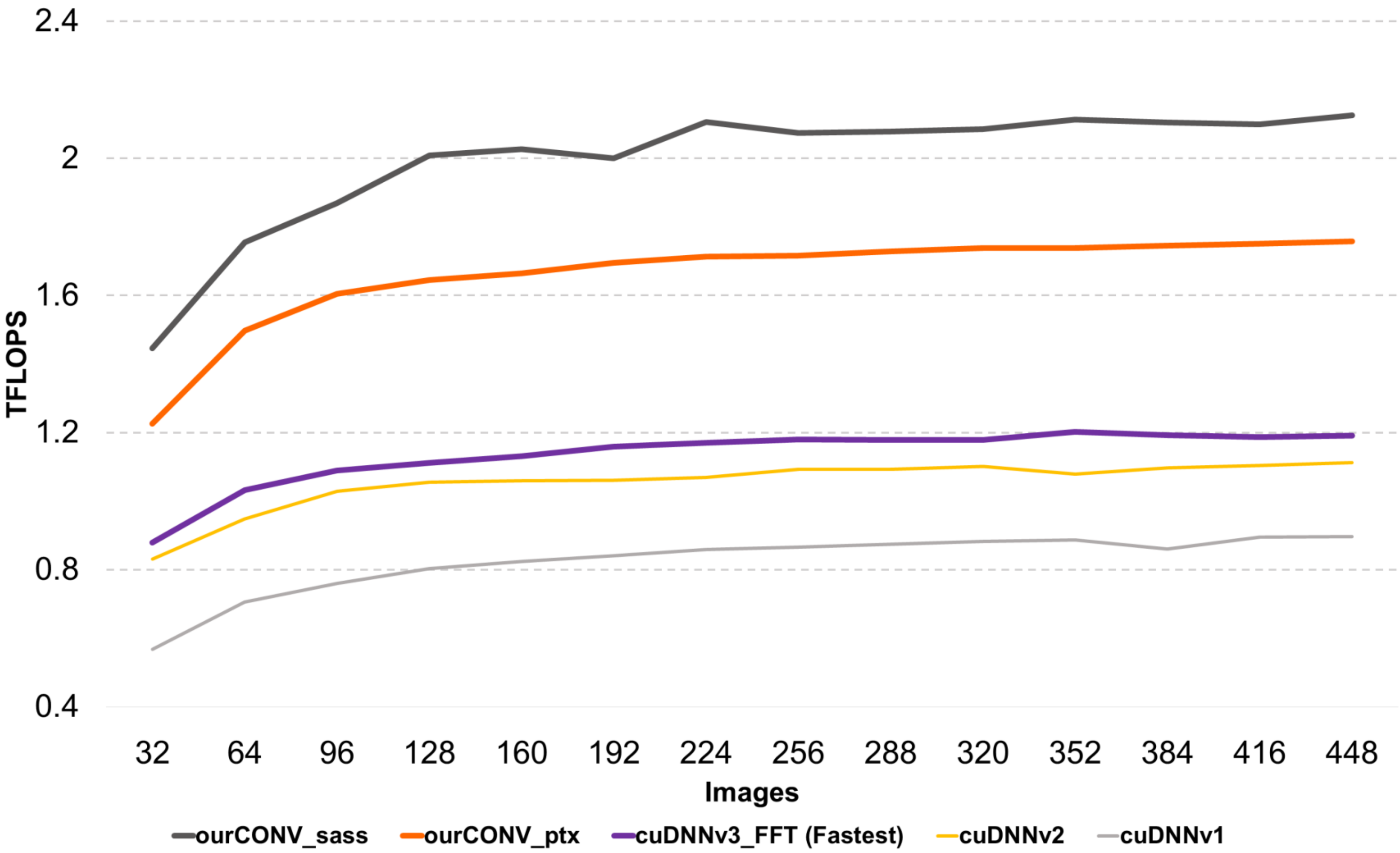
Tuning Performance on Kepler GPUs

An Introduction to Kepler Assembler and Its Usage in CNN optimization

Cheng Wang, Zhe Jia, Kai Chen

Alibaba

Convolution Layer Performance on K40



Height = 16; Width = 16; Channel = 5; Stride = 1; Ksize = 5; Pad = 2; Neuron = 32

What will you learn?

Part I: The technologies to improve performance.

- ✓ High level optimizations
- ✓ Low level optimizations

.

Part II: The tool to achieve optimizations from Low Level.

- ✓ How to use Kepler assembler
- ✓ Some tips about performance

Part I: The Technologies

Sometimes, you have to write kernels by yourselves!

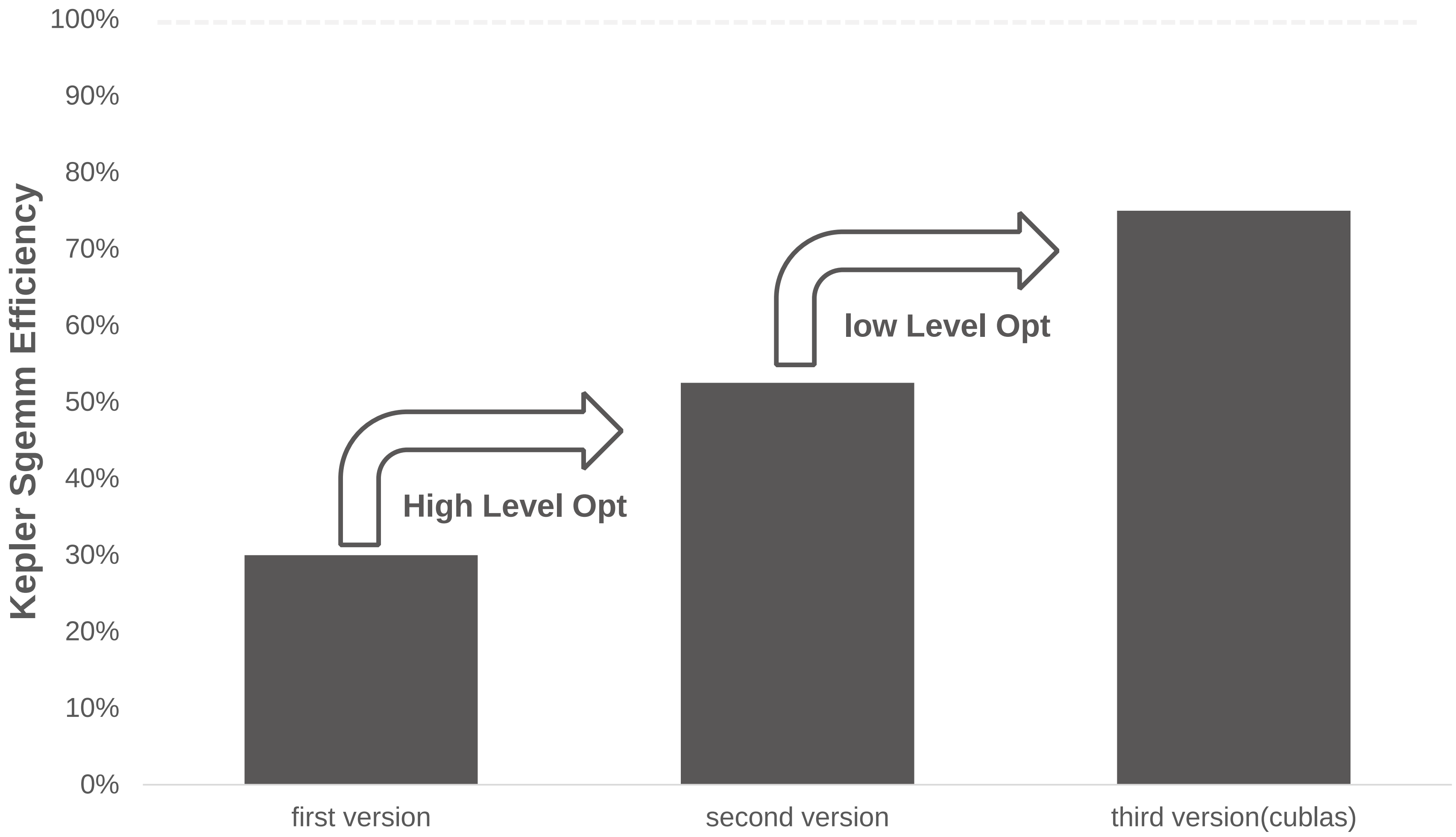
- Kernel launch time is expensive. Combine different kernels;

  Example: If m, n and k are very small in Sgemm, GPU computing time will be short.
- Global memory access is expensive. Should be minimized;

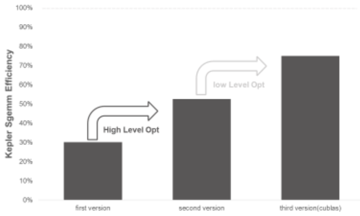
  Example: im2col+sgemm = direct convolution implementation
- CUDA library cannot always fulfill your performance requests;

  Example: low latency? High throughput?

Different kinds of optimizations

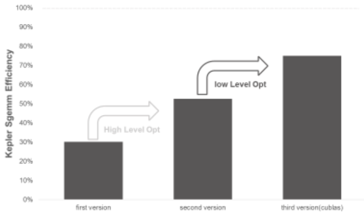


Different kinds of optimizations-High Level



- Minimize data transfer between the host and the device;
- Coalesce global memory accesses;
- Minimize use of global memory, and use shared memory to reduce global memory access;
- Avoid different execution paths within the same warp (divergence);
- Occupancy (thread/block management, shared memory limitation);

Different kinds of optimizations-Low Level



- Use more vectorized instructions:

  LDS.128, STS.128;
- Use more dual issues:

  two dispatchers issue sequential instructions at same clock;
- Schedule instructions efficiently, get better ILP:

  reorder instructions, hide long latency memory access instructions;
- Occupancy(register availability):

  register number, register re-use;

BUT it is hard to achieve most of low level optimizations

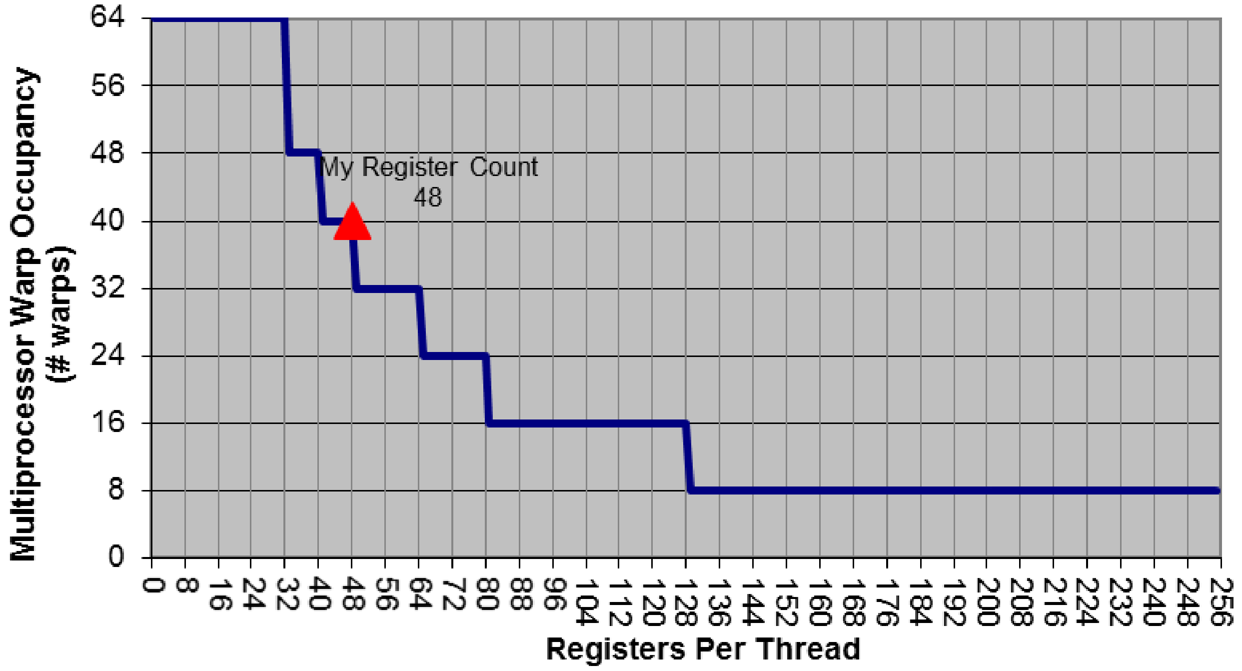
Our Experiences

Compiled by nvcc, ~70% performance of cuBLAS Sgemm with exact same algorithm.

Reasons:

1. Low Occupancy:

   too many registers.



BUT it is hard to achieve most of low level optimizations

Our Experiences

Compiled by nvcc, ~70% performance of cuBLAS Sgemm with exact same algorithm.

Reasons:

1. Low Occupancy:

   too many registers.

“-maxrregcount n”?

register spilling to local memory!

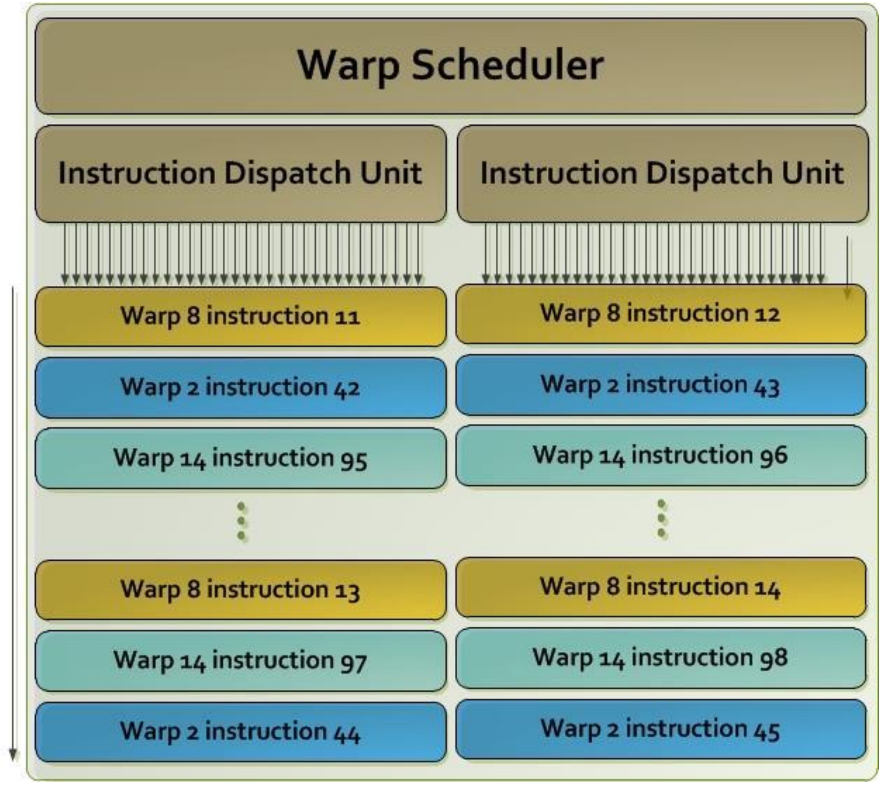
BUT it is hard to achieve most of low level optimizations

Our Experiences

Reasons:

2. Bad Instruction Level Parallelism(ILP)

```
                                             /* 0x0880101410141014 */
/*0f08*/   FFMA R37, R88, R86, R37;         /* 0xcc0094002b1d6096 */
/*0f10*/   FFMA R34, R89, R86, R34;         /* 0xcc0088002b1d648a */
/*0f18*/   FFMA R46, R89, R87, R46;         /* 0xcc00b8002b9d64ba */
/*0f20*/   FFMA R39, R88, R92, R39;         /* 0xcc009c002e1d609e */
/*0f28*/   FFMA R35, R88, R87, R35;         /* 0xcc008c002b9d608e */
/*0f30*/   FFMA R40, R89, R92, R40;         /* 0xcc00a0002e1d64a2 */
/*0f38*/   IADD R108.CC, R108, R119;        /* 0xe08400003b9db1b2 */
```



Part II: The Tool

Kepler Assembler

Kepler Assembler

xxx.cubin

```
0x088cb0a0a08c1000
0x64c03c00089c0006
0x8640000109c000e
0x74000000021fc016
0x8640000129c0002
0x51080c00051c000a
0x910c1400281c081a
0x93181400289c081e
```

xxx.sass

```
CTL:00000000        MOV R1, c[0x0][0x44];
CTL:00000100        S2R R0, SR_CTAID.X;
CTL:00100011        MOV32I R5, 0x4;
CTL:00101000        S2R R3, SR_TID.X;
CTL:00101000        IMAD R2, R0, c[0x0][0x28], R3;
CTL:00101100        IMAD R6.CC, R2, R5, c[0x0][0x140];
CTL:00100011        IMAD.HI.X R7, R2, R5, c[0x0][0x144];
```

Kepler Assembler

GPU Architecture

```
code for sm35
```

Kernels in this cubin file

```
Kernel number: 14
```

Current kernel name

```
<KernelName: sgemm_sm35_ldg_nn_128x8x128x16x16
```

Parameter info

```
<Para__sgemm_sm35_ldg_nn_128x8x128x16x16: num|3 size|24
<ParaDetail__sgemm_sm35_ldg_nn_128x8x128x16x16:
        Index  |  Addr  |  Size  |  Align
          1    | 0x140  |   8    |    0
          2    | 0x148  |   8    |    0
          3    | 0x150  |   8    |    0
ParaDetail>
```

Shared mem info

```
<Shared__sgemm_sm35_ldg_nn_128x8x128x16x16: size|8340 align|4
```

Register number

```
<Reg__sgemm_sm35_ldg_nn_128x8x128x16x16: 127
```

Control code and Instruction

```
<code:
        /*0008*/CTL: 00100011    S2R R96, SR_TID.X;
        /*0010*/CTL: 00100000    S2R R110, SR_TID.Y;
        /*0018*/CTL: 00100000    ISCADD R98, R110, R96, 0x4;
```

Use Kepler Assembler

A. Generate Cubin(demo.cubin)

demo.cu:

```
__global__ void kernel(float* array1, float* array2, float* array3){
        int tid = threadIdx.x;
        int bid = blockIdx.x;
        int offset = tid+bid*blockDim.x;

        array3[offset] = array1[offset]*array2[offset];
}
```

```
$nvcc -cubin -gencode arch=compute_35,code=sm_35 demo.cu
```

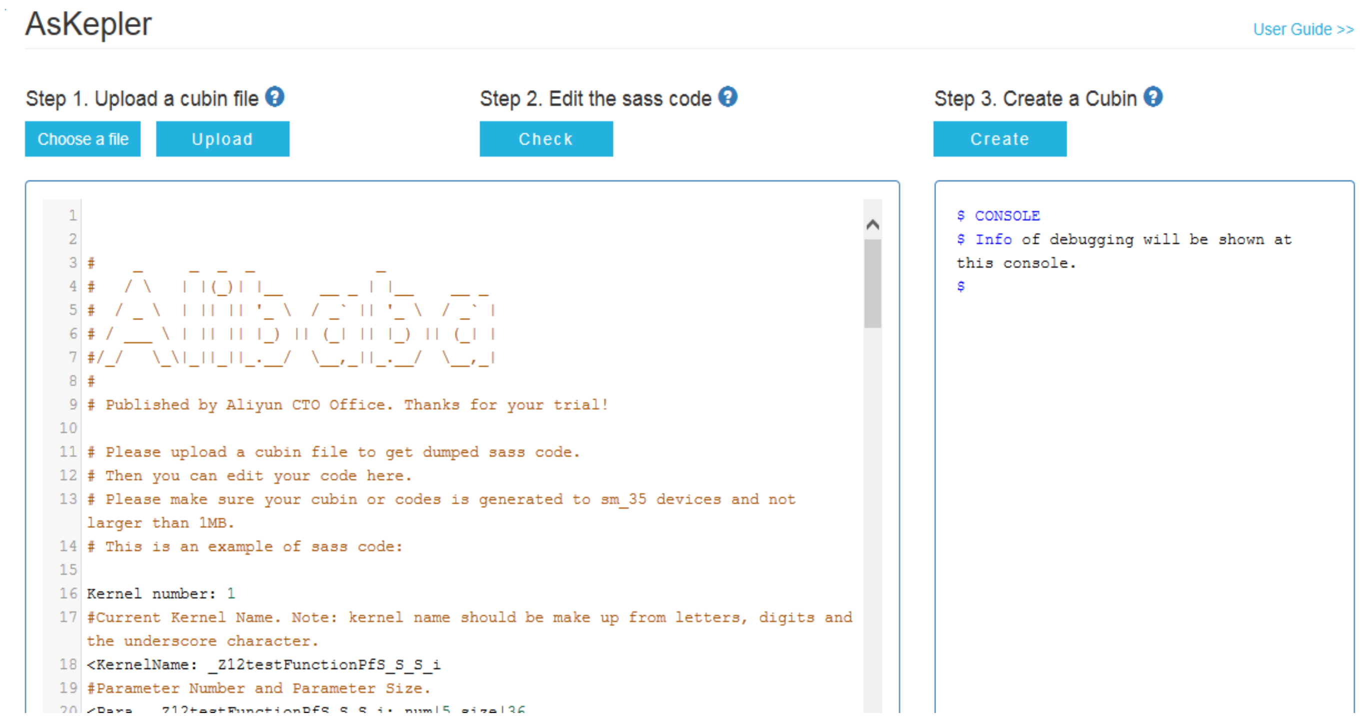
Use Kepler Assembler

B. Optimize Cubin

Upload cubin into AsKepler

Edit SASS and Optimize.

Generate cubin



Use Kepler Assembler

C. Use cubin in your code

CUDA module:

```
// load Module from cuda binary file
cuModuleLoad(&cuModule, “demo.cu.cubin");

// get kernel from Module
cuModuleGetFunction(&mykernel, cuModule, “kernelname");

// launch kernel
cuLaunchKernel(kernelname,  GridDim.x, GridDim.y, GridDim.z,
               BlockDim.x, BlockDim.y, BlockDim.z,
               sharedMemBytes,
               stream, args, hStream);
```

Use Kepler Assembler

D. Tips about performance:

- Use as many “real” dual issues as you can;
- Understand the meaning of control code;
- Be careful with register bank conflict;
- Take a look at “special control code” used in cuBLAS kernel(pay attention to instruction blocks with more than 3 FFMA instructions);

Current Work

- Automatic Gemm and CNN kernel generator.

- Performance tuning for our clients

AliCloud-HPC	G2	G4
CPU	Intel Xeon E5 v2 CPU (x2)	Intel Xeon E5 v4 CPU (x2)
GPU	Tesla K40 (x2)	Tesla M40 (x2)
Mem	128GB DDR3	128GB DDR4
Storage	2TB HHD (x8)	1.92TB SSD (x8)
Theoretical Peak (SP)	~11 TFLOPs	~16 TFLOPs

Thank you!

Contact info:

About CUDA kernel performance and Kepler Assembler:

Zhe Jia jiazhe.jz@alibaba-inc.com

Kai Chen kevinchen.ck@alibaba-inc.com

About HPC GPU Server Purchase:

Cheng Wang changren@taobao.com

Use Kepler Assembler (Free):

1 visit https://hpc.aliyun.com

2 click “En” at upper right corner to change into English page

3 “Product & Service”->”tools”->”AsKelper”

4 finish register process, and login

5 visit https://hpc.aliyun.com, “Product & Service”->”tools”->”AsKelper”