GTCx

SEOUL | Oct.7, 2016

금융회사 CUDA 활용 방안

- 보험 계리 프로그램 개선 개발

라이칸㈜ 대표이사 김영길, 2016-10-07

PRESENTED BY

NVIDIA.

AGENDA

1. Project 개요
2. Project 수행 내역
3. 향후 개선/연구 계획

1. PROJECT 개요

PROJECT 발생 이유 - WHY?

외부환경 변화	대응책 필요
보험사의 보증준비금 산출 시 외부 변수(수익률 시나리오 수 등)의 급증	기존 사용 계리소프트웨어의 한계
외부 변수의 지속적인 증가 예상	지속적인 장비 증설의 한계
신기술의 탄생(GPU)	기존과 같은 대응방식을 통해서는 현 수준을 유지하는 것도 어려움

PROJECT 목표 - TARGET!

목표	결론
CPU 대비 6x 속도 향상	CPU 대비 약22배 속도 향상
데이터 급증 시 또는 더욱 속도를 향상 시키기 위하여 소스코드의 전면 수정이 아닌 설정을 통하여 퍼포먼스를 획기적으로 향상	멀티 GPU 사용으로 GPU 연산 부분을 1/GPUs로 감소시킴

업무 개요 - WHAT!

Case : 5가지(Default 포함)

Mortality		사망률	
		-x %	+y %
해지율	+j %		
	-k %		

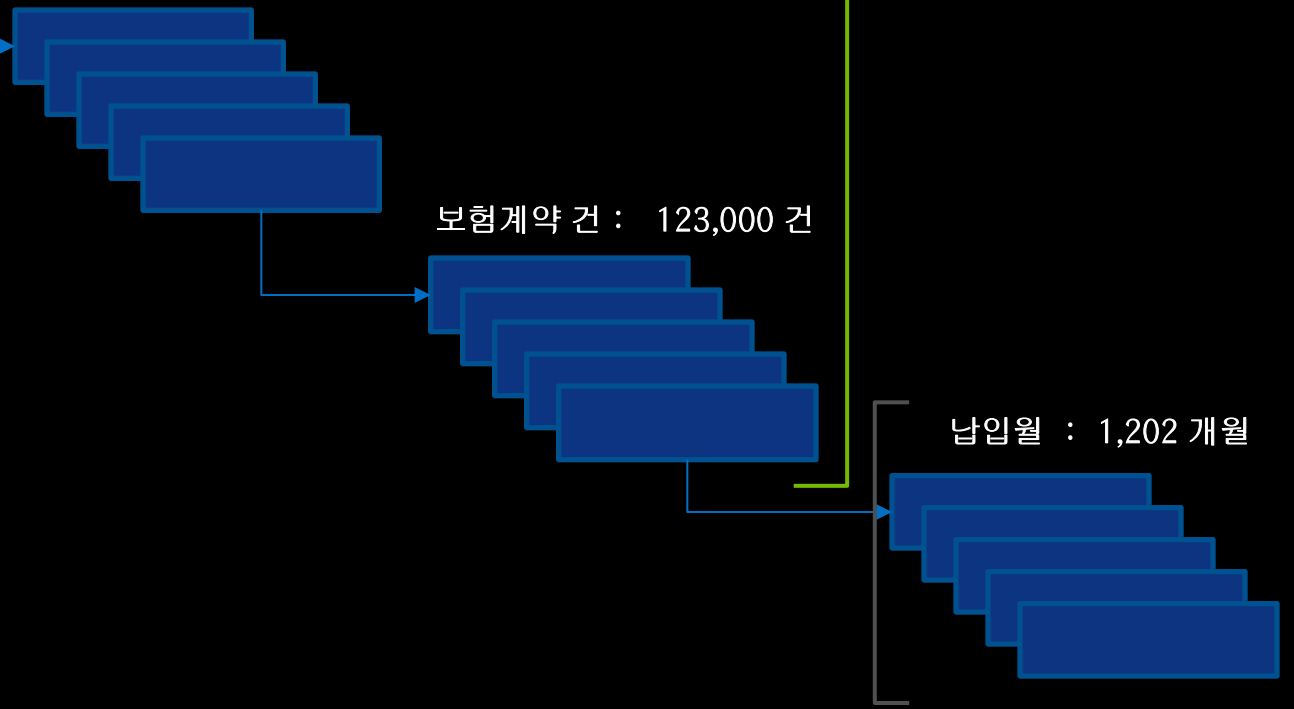
수익률 시나리오 : 1,000개

보험계약 건 : 123,000 건

납입월 : 1,202 개월

모두 병렬화 가능

병렬화 불가



2. PROJECT 수행 내역

GTCx

개발 환경
Dev Environment

개발시스템 사양

OS : Windows7 64bit

CPU : Intel Xeon E3-1225 V3

GPU : NVIDIA Tesla k40c 1EA

RAM : 16GB

기초데이터

보증준비금 산출 로직 Excel : 실제 계산식과 검증 가능한 값이 포함되어 있는 파일

기타데이터 : 수익률 시나리오, 보험 계약정보, 각종 수수료 및 기타 Input 정보

병렬화 과정

보증준비금 산출 로직 Excel 자료를 기초로 한 C코드 개발 및 값 검증(시간 측정)

병렬화 아키텍쳐 수립

병렬화(CUDA) -> 최적화->값 검증(시간 측정)

CPU 버전 개발

CPU Main Module

- 계산 입력파일 처리부
  - STL의 map과 list를 이용,
    - –문자열 분리의 원활한 처리를 위해서 getline을 이용
    - –수익률 시나리오데이터의 경우 접근속도 향상을 위해 double 1차원 배열로 유지
- 보험계약정보(약12만 건)별 연산 분기
  - 단일/다중 펀드 및 추가납 상황으로 분기처리
- 수익률 시나리오(500*2) 및 비율(사망률/해지율의 5개 조건 상황) 적용 연산부
- 출력파일 처리부
  - 보험계약정보 묶음별 CSV 파일로 결과 출력.

```
void CPOK_GR_GPU2Dlg::ProcessVerifyMode()
{
    m_clsScenario.ReadData(...);
    ...
    m_lstDied=(double*) new double [rate_max*scen_max*month_max];
    ...
    for(map<string, void*>::iterator itrPMF=m_clsPMF.m_lstCondition.begin(); itrPMF!=m_clsPMF.m_lstCondition.end(); itrPMF++)
    {
        pContract=(stCondition*)itrPMF->second;
        clsCOI.ReadData((LPCTSTR)strFilePath);
        ...
        clsCOI.ToArray(pRateInfo[eRateInfo_COI]);
        ...
        PMF::FindSnrVectorName(pContract->lstSnrIndex, pContract->lstFund, m_clsScenario.m_mpKey);
        ...
        bIsAP=(pContract->lstBase[UM_PLAN_CODE].find("AP")!=std::string::npos);
        if(pContract->NumberOfFunds<=1){
            if(bIsAP) runSingleAP(...);
            else runSingle(...);
        } else {
            if(bIsAP) runMultipleAP(...);
            else  runMultiple(...);
        }
        ...
        WriteVerifyResult(sPolicyNo.c_str(), m_clsScenario, cMode[result_idx], nPMFIndex, month_max);
        ...
        clsCOI.ReleaseData();
        ...
    }
    delete [] m_lstDied;
    ...
    delete [] pRateInfo[eRateInfo_COI];
}
```

CPU 버전 개발

CPU Main Module Details

```
void CPOK_GR_GPU2Dlg::ProcessVerifyMode()
{
    m_clsScenario.ReadData(...);
    ...
    m_lstDied=(double*) new double [rate_max*scen_max*month_max];
    ...
    for(map<string, void*>::iterator itrPMF=m_clsPMF.m_lstCondition.begin(); itrPMF!=m_clsPMF.m_lstCondition.end(); itrPMF++)
    {
        pContract=(stCondition*)itrPMF->second;
        clsCOI.ReadData((LPCTSTR)strFilePath);
        ...
        clsCOI.ToArray(pRateInfo[eRateInfo_COI]);
        ...
        PMF::FindSnrVectorName(pContract->lstSnrIndex, pContract->lstFund, m_clsScenario.m_mpKey);
        ...
        bIsAP=(pContract->lstBase[UM_PLAN_CODE].find("AP")!=std::string::npos);
        if(pContract->NumberOfFunds<=1){
            if(bIsAP) runSingleAP(...);
            else runSingle(...);
        } else {
            if(bIsAP) runMultipleAP(...);
            else  runMultiple(...);
        }
        ...
        WriteVerifyResult(sPolicyNo.c_str(), m_clsScenario, cMode[result_idx], nPMFIndex, month_max);
        ...
        clsCOI.ReleaseData();
        ...
    }
    delete [] m_lstDied;
    ...
    delete [] pRateInfo[eRateInfo_COI];
}
```

수익률 시나리오 및 모듈 정보
파일 데이터 읽기

보험계약정보 별 Looping(123,000건)

수집된 비율 리스트를 1차원 배열로 치환

보험 계약정보와 연계되는
수익률 시나리오의 경과월별 백터값을 추출

CPU 버전 개발

CPU Core Module

```
void kernel_single(int *pmf, double *SnrData, ..., date tProjStart, ..., double *lstDied, ...)
{
    for(int no_rate=0; no_rate<RATE_MAXCOUNT; no_rate++)
    {
        if(no_rate==0) { fDieRate=0.95; fCancelRate=0.9; break;}
        ...
        for(int scenario=0; scenario<scen_max; scenario++)
        {
            nResultIdx=no_rate*snrNoCount*snrMonthlyCount+no_idx*snrMonthlyCount;
            for(month_idx=0; month_idx<snrMonthlyCount; month_idx++)
            {
                if(fPrevReserved<=MIN_VALUE) fCancelled=0.0;
                else {
                    ...
                    fCancelled=(1.0-fCancelled) * fPrevMaintained;
                }
                ...
                lstDied[nResultIdx+month_idx]=fDied;
                ...
                fPrevReserved=fReserved;
                ...
            }   //end-monthly
        }   //end-snr_no
    }   //end-rate
}
```

- 사망자율/해지율 적용에 따른 Loop : 5
- 수익률 시나리오 Loop : 1,000
- 경과월 수 Loop : 1,202
- 다음 경과월에 대한 참조값 저장 (병렬화 방해 요인)

GPU 버전 개발 1

CUDA Main Module

```
void CPOK_GR_GPU2Dlg::ProcessVerifyMode()
{
    cudaSetDeviceFlags( cudaDeviceMapHost );
    ...
    while(bFoundFile)
    {
        bFoundFile=findFile.FindNextFile();
        arrFilePath.Add(findFile.GetFilePath(), findFile.GetFileTitle(), findFile.GetLength());
    }
    arrFilePath.Sort();
    ...
    cudaError_t err=cudaHostAlloc((void**)&m_clsScenario.m_dwSnrData,
                sizeof(double)*SNR_MAX*SNR_TOTALCOUNT, cudaHostAllocWriteCombined|cudaHostAllocMapped);
#pragma omp parallel for
    for(int fileno=0; fileno<filecount; fileno++)
    {
        char *buf=new char[arrFilePath.GetFileSize(fileno)+1];
        f=fopen(strFilePath, "r");
        while((read=fread(buf[fileno]+nTotalRead, 1, 1024, f))>0) nTotalRead+=read;
        buf[fileno][nTotalRead]=0;
        m_clsScenario.ReadData(fileno, buf);
        ...
    }
    for (int fileno = 0; fileno<count; fileno++)
    {
        runCsvToDbl(buf[fileno], pSnrPos[fileno],
                    xy_count, m_clsScenario.m_dwSnrData + fileno*block_size, block_size);
    }
    ...
    m_clsPMF.ReadInfoFile(m_strInfoRoot, m_strLoadRoot, m_strCOIRoot, m_strLapseRoot, m_strMortRoot);
    ...
    bFoundFile=findFile.FindFile(m_strPMFRoot+_T("\\*.VF.csv"));
    while(bFoundFile){
        bFoundFile=findFile.FindNextFile();
        m_clsPMF.ReadFundData((LPCTSTR)findFile.GetFilePath(), clsReserved1);
        ...
    }
    ...
    bFoundFile=findFile.FindFile(m_strPMFRoot+_T("\\*.00.csv"));
    while(bFoundFile){
        bFoundFile=findFile.FindNextFile();
        m_clsPMF.ReadPolicyData((LPCTSTR)findFile.GetFilePath());
        ...
    }
    ...
    if(bSingleMode)    runSinglePMF_gpu(...);
    if(bSingleAPMode) runSingleApPMF_gpu(...);
    if(bMultiMode)     runMultiplePMF_gpu(...);
    if(bMultiAPMode)   runMultipleApPMF_gpu(...);
    ...
    clsReserved1.m_mpData.clear();
    ...
    cudaDeviceReset();
}
```

- Mapped Memory 사용을 위한 플래그 설정
- 수익률 시나리오 관련 파일명 및 크기 정보 저장
- Mapped Memory 로 수익률 시나리오 데이터 할당
- OpenMP를 통하여 병렬화 하여 수익률 시나리오 파일을 읽어들임
- 수치형문자값(1,000 X 1,202)을 CUDA에서 double형으로 변환(병렬처리)
- 펀드유형별로 CUDA 커널함수 호출
  (모든 보험계약정보는 커널호출 함수 내부에서 블럭화되어 병렬처리 됨)

GPU 버전 개발 2

GPU Kernel Call Module

```
int runSinglePMF_omp(..., PMF *pPmf, double *ppSnrData, ...)
{
    cudaHostAlloc((void**)&h_Died, sizeof(double)*mem_size, cudaHostAllocWriteCombined|cudaHostAllocMapped);
    cudaHostGetDevicePointer((void**)&d_oDied, (void*)h_Died, 0);
    ...
    cudaHostGetDevicePointer((void**)&d_SnrData2, (void*)ppSnrData, 0);
    ...
    cudaMalloc((void**)&d_LapseRate2, mem_size);
    cudaMemcpy(d_LapseRate2, pPmf->m_pLapse, mem_size, cudaMemcpyHostToDevice);
    ...
    for(map<int, list<string>>::iterator itrSingle=pPmf->m_mpSingle.begin(); itrSingle!=pPmf->m_mpSingle.end(); itrSingle++)
    {
        pmf_sub_count=pmf_size/nPmfOnBlock;
        itrPMF=itrSingle->second.begin();
        for(int pmf_sub=0; pmf_sub<pmf_sub_count; pmf_sub++)
        {
            for( ; itrPMF!=itrSingle->second.end() && pmf_idx<pmf_sub_size; itrPMF++, pmf_idx++, mem_size+=PMF_MODELSIZE)
            {
                pContract=pPmf->m_lstCondition.find(*itrPMF)->second;
                ...
                pmf2[pmf_idx*ePMF_MAX+ePMF_PREM_CODE]=atoi(pContract->lstBase[PREM_CODE].c_str());
                ...
                pPmf->GetCOIData(pContract, pCOI+mem_size, nRateInfoCount[pmf_idx*eRateInfo_Max+eRateInfo_COI]);
                ...
            }
            cudaMemcpy(d_COIMortRate2, pCOI, mem_size, cudaMemcpyHostToDevice);
            ...
            nBlockOnGrid=(pmf_sub_size+nThreadOnBlock-1)/nThreadOnBlock;
            ...
            kernel_days360_s<<<dim3(nBlockOnGrid, SNR_MONTHLY_MAXCOUNT, 1), nThreadOnBlock>>>(...);
            kernel_die_s<<<dim3(nBlockOnGrid, DIE_MONTH_COUNT, 1), nThreadOnBlock>>>(...);
            kernel_cancel_s<<<dim3(nBlockOnGrid, CANCEL_MONTH_COUNT, 1), nThreadOnBlock>>>(...);
            ...
            kernel_pmf_single<<<dim3(nBlockOnGrid, RATE_SNR_COUNT, 1), nThreadOnBlock>>>(...);
#pragma omp parallel num_threads(threadCount)
            {
                list<string>::iterator omp_pmf=itrSingle->second.begin();
                ...
#pragma omp for  reduction(+: dwDied, ...)
                for(int pmf_idx=0; pmf_idx<pmf_sub_size; pmf_idx++)
                {
                    int thread_idx=omp_get_thread_num();
                    if((omp_idx%nVerifyDataSize)==0){
                        if(bOpenedOmpFile) fclose(hOmpFile);
                        sprintf_s(pszOmpFilePath, "%s(%d).csv", pszVerifyFileName, verify_idx+pmf_idx+1);
                        hOmpFile=fopen(pszOmpFilePath, "w");
                        WriteVerifyHeaderText(hOmpFile);
                    }
```

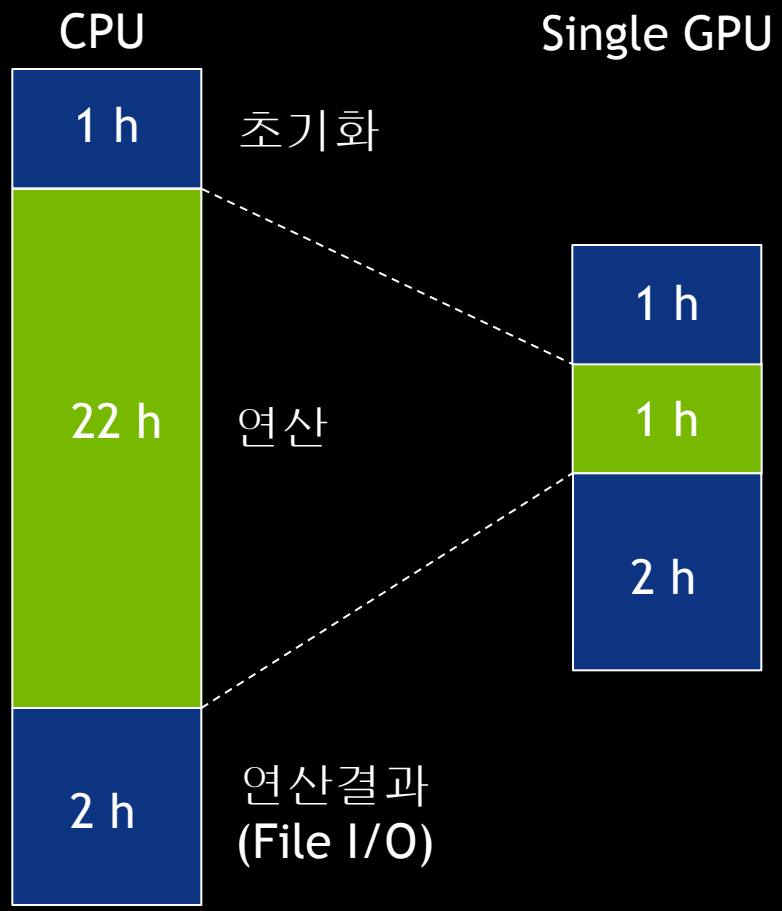
- 결과 호스트 변수 할당
- 호스트변수와 디바이스변수 연결(zero copy)
- 지정된 경과월 수 크기별 블럭단위 Looping
- 커널의 블록 수 결정
- 분리 가능한 계산식 커널
- 산출식 커널
- OpenMP를 통한 결과값 병렬 출력

최종 비교

GPU vs CPU

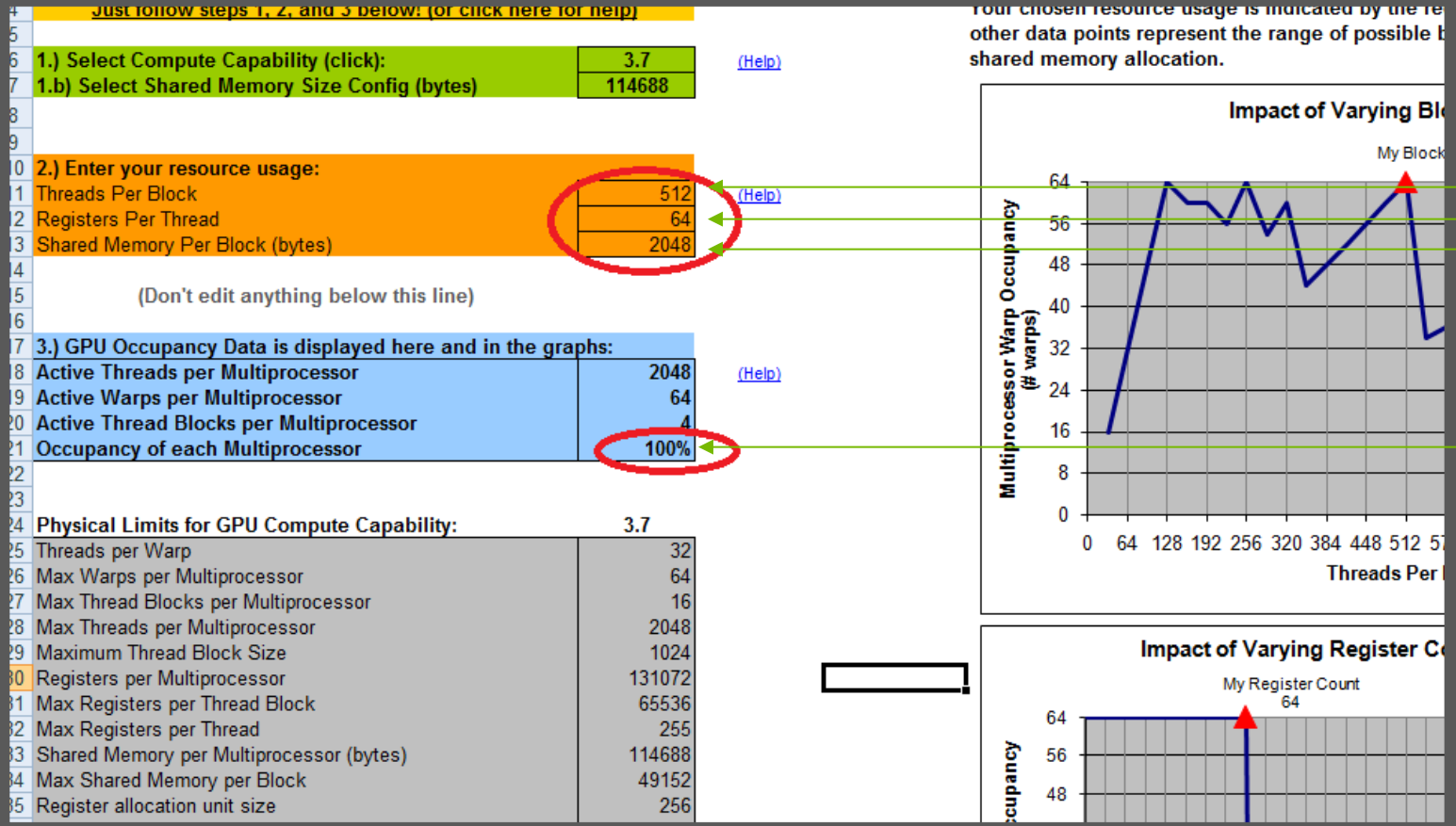
1. 테스트 결과(동일 machine)
- 총 수행 시간 : CPU 25h / GPU 4h(약6배)
- 연산 부분 : CPU 22h / GPU 1h(약22배)
- Double precision 연산

2. 시사점
- 연산에 들어가는 시간의 약 20배 속도 향상
- 향후 연산 부분의 환경 변화 시, GPU 1장 추가로 현 환경의 2배를 동일한 시간에 처리 가능함.
- Single precision으로 계산 시 약 3배의 속도 추가 향상 가능
- GPU 병렬화 + CPU 병렬화를 동시에 수행함으로 최적의 퍼포먼스를 보장할 수 있음.



최적화 TIP 1

CUDA_Occupancy_Calculator 사용



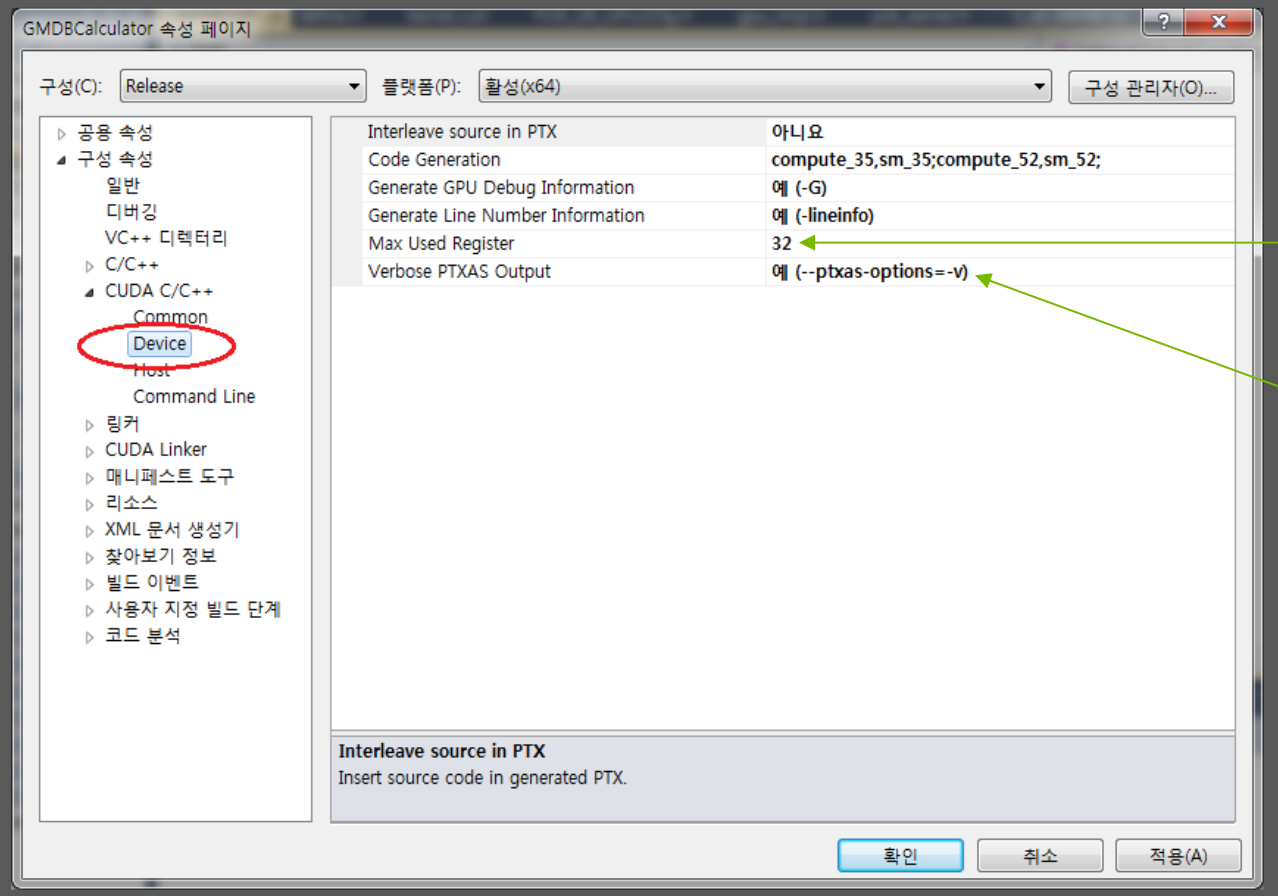
Just follow steps 1, 2, and 3 below! (or click here for help)		
1.) Select Compute Capability (click):	3.7	(Help)
1.b) Select Shared Memory Size Config (bytes)	114688	
2.) Enter your resource usage:		
Threads Per Block	512	(Help)
Registers Per Thread	64	
Shared Memory Per Block (bytes)	2048	
(Don't edit anything below this line)		
3.) GPU Occupancy Data is displayed here and in the graphs:		
Active Threads per Multiprocessor	2048	(Help)
Active Warps per Multiprocessor	64	
Active Thread Blocks per Multiprocessor	4	
Occupancy of each Multiprocessor	100%	
Physical Limits for GPU Compute Capability:	3.7	
Threads per Warp	32	
Max Warps per Multiprocessor	64	
Max Thread Blocks per Multiprocessor	16	
Max Threads per Multiprocessor	2048	
Maximum Thread Block Size	1024	
Registers per Multiprocessor	131072	
Max Registers per Thread Block	65536	
Max Registers per Thread	255	
Shared Memory per Multiprocessor (bytes)	114688	
Max Shared Memory per Block	49152	
Register allocation unit size	256	

블록당 할당할 스레드 수
스레드 당 사용할 레지스터 수
블록내에 사용할 공유메모리 크기

위 값을 조정하여 Occupancy가 100%에 근접할 수 있도록 조정

최적화 TIP 2

Build Option



커널에서 사용하는 스레드 당 레지스터의 수를 제한하여 블록당 가용한 스레드를 조정

```
ptxas info    : Function properties for kernel_days360_s
    16 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
ptxas info    : Used 47 registers, 16 bytes cumulative stack size, 392 bytes cmem[0]
ptxas info    : Function properties for cudaMalloc
    16 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads
```

빌드 후 메모리 사용량 등을 확인하여 레지스터 메모리가 지역 메모리에 설정되어진 양을 확인하여 메모리 속도 지연 감소를 최소화하기 위한 옵션

최적화 TIP 3

Multi GPU

```
2     int nDeviceCount(0);
3     cudaGetDeviceCount(&nDeviceCount);
4  #pragma omp parallel num_threads(nDeviceCount)
5  {
6  #pragma omp for
7     for(int pmf_idx=0; pmf_idx<nPMFCount; pmf_idx++, itrSingle++)
8     {
9        int cpu_thread_id=omp_get_thread_num();
10       cudaSetDevice(cpu_thread_id);
11       cudaSetDeviceFlags(cudaDeviceMapHost);
12       ...
13    }
14    ...
15 }
```

- GPU Count
- OpenMP의 스레드 총 수를 GPU 수로 할당
- 현재 작업 스레드 번호
- 스레드 번호에 맵핑되는 장치 ID를 설정

GTCx

최적화 TIP 4

최적화 Tip 기타

계약정보 BLOCK화

경과 월수에 따른 블록화
보험계약정보 블록화

연산에 대한 수식 개선

Constant, shared, zero copy
다중계산식의 단항식화
조건분기문 소거 및 개선

계산식 분리

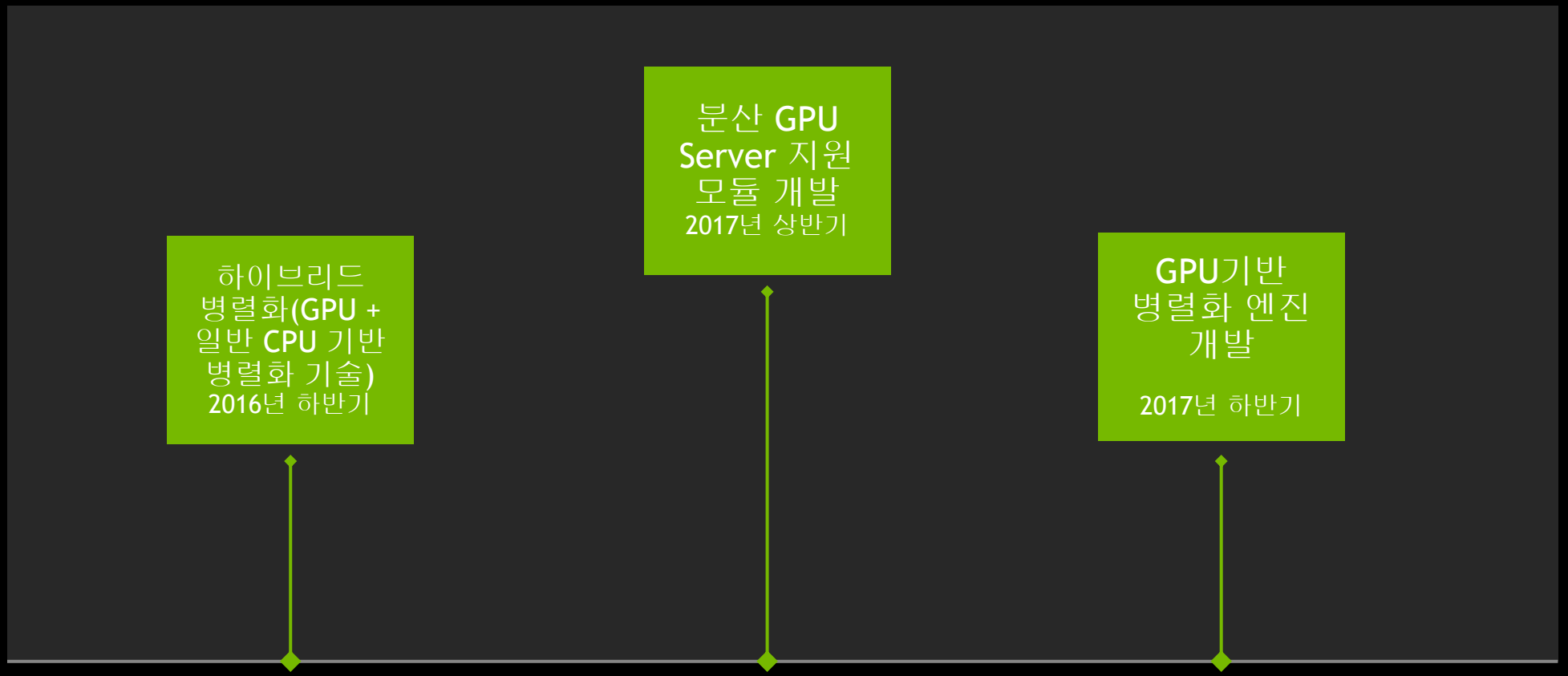
독립처리 가능 부분 분리
선처리 가능 부분 분리

OPENMP

Multi-GPU 활용
CPU 부분 선,후처리 병렬화

3. 향후 개선/연구계획

TIMELINE



Source: Source information is 8 pt, italic

"금융시장은 미래예측이 그 어느 때 보다 중요한 시점이며, 이를 위해서는 다양한 시나리오를 바탕으로 수많은 예상치를 연산하는 능력이 중요함으로 향후 금융영역에서 **GPU** 및 **CUDA**가 중요한 위치를 차지 할 것입니다"

– 금융시장에서의 **GPU** 및 **CUDA**의 역할

GTCx

SEOUL | Oct.7, 2016

THANK YOU

JOIN THE CONVERSATION

#GTCxKorea2016

PRESENTED BY

NVIDIA