Transparent Offloading and Mapping (TOM)

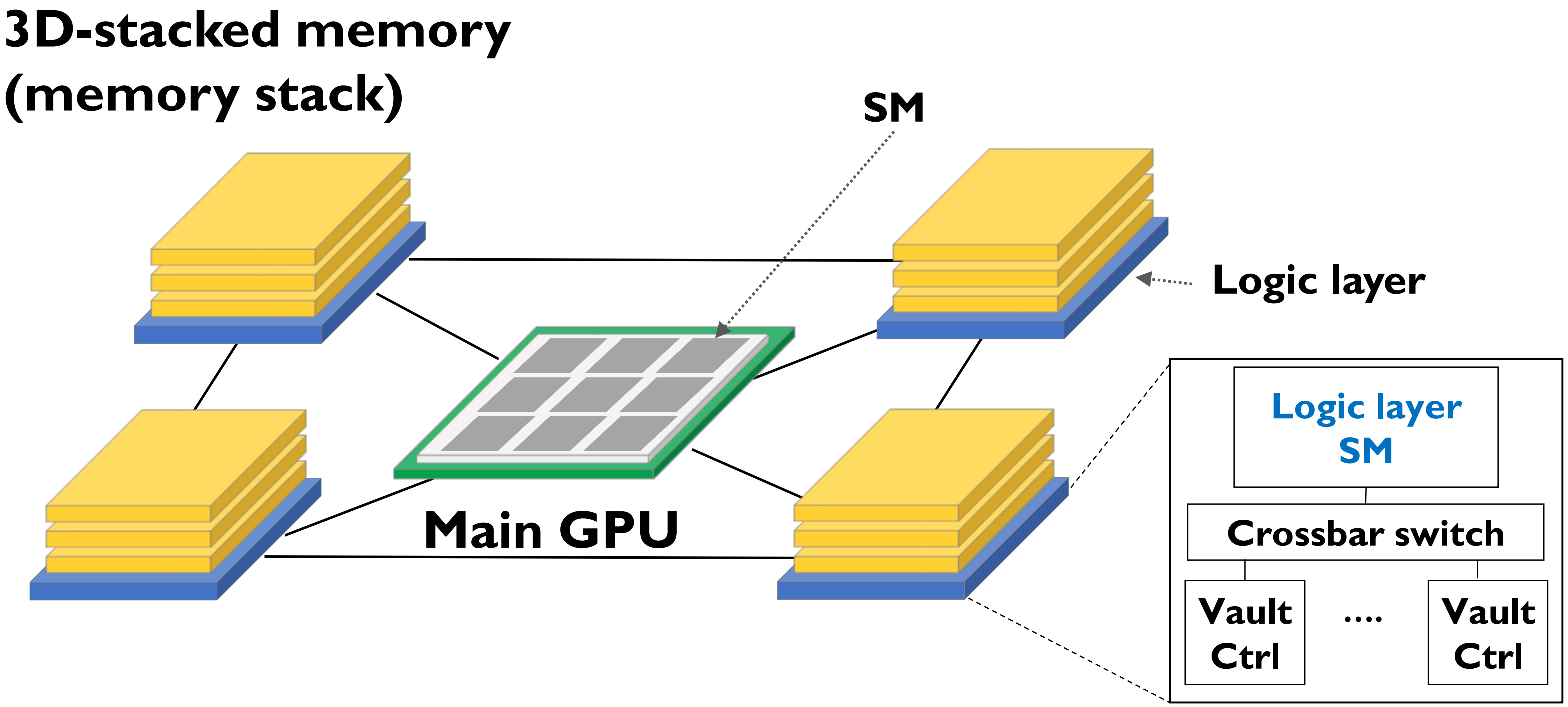
Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh

Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee,
Mike O’Connor, Nandita Vijaykumar,
Onur Mutlu, Stephen W. Keckler

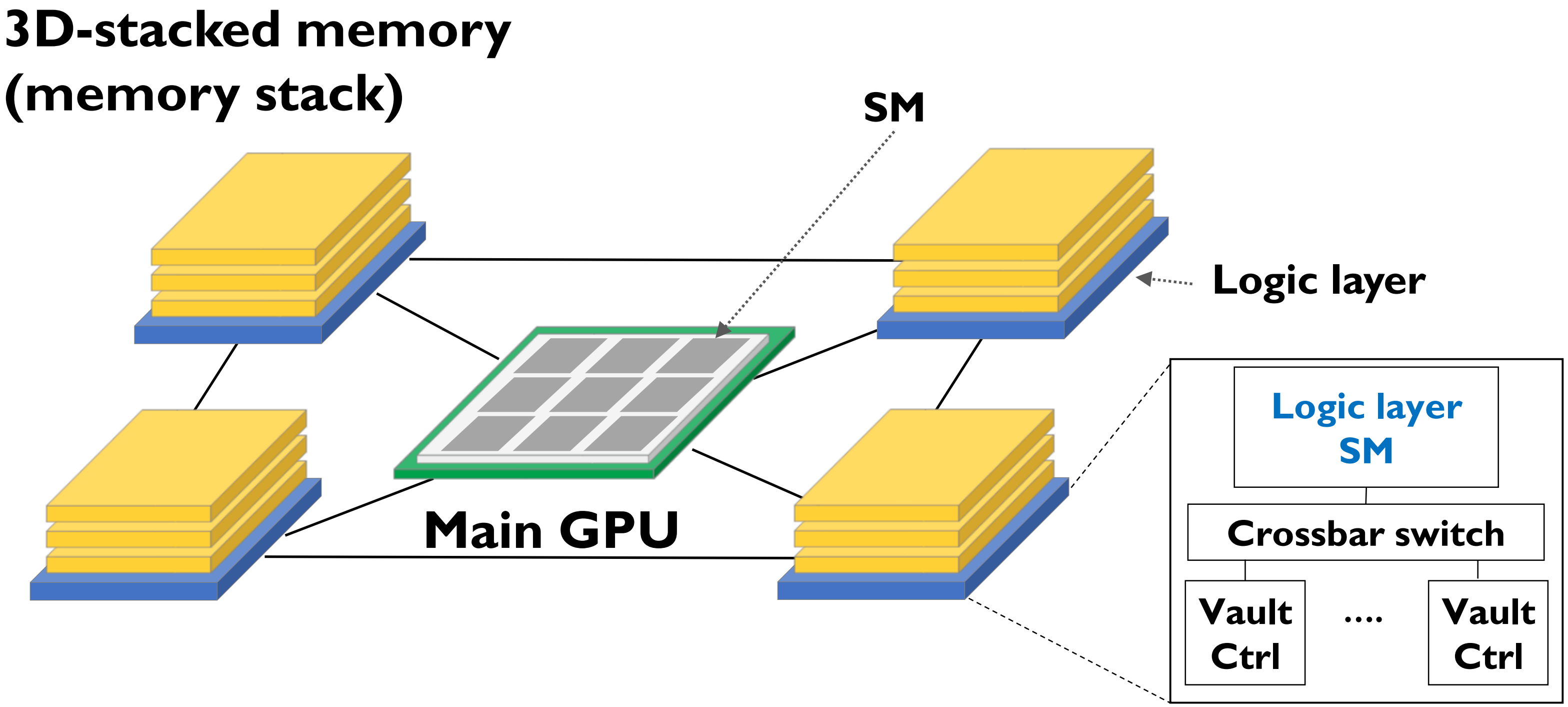
SAFARI Carnegie Mellon NVIDIA KAIST ETH zürich

Motivation



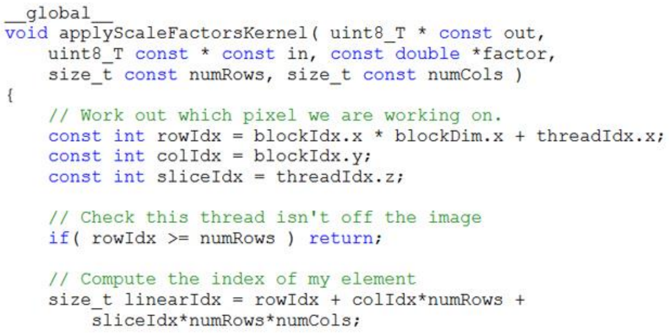
Processing data directly in 3D-stacked memories is a promising direction

Motivation

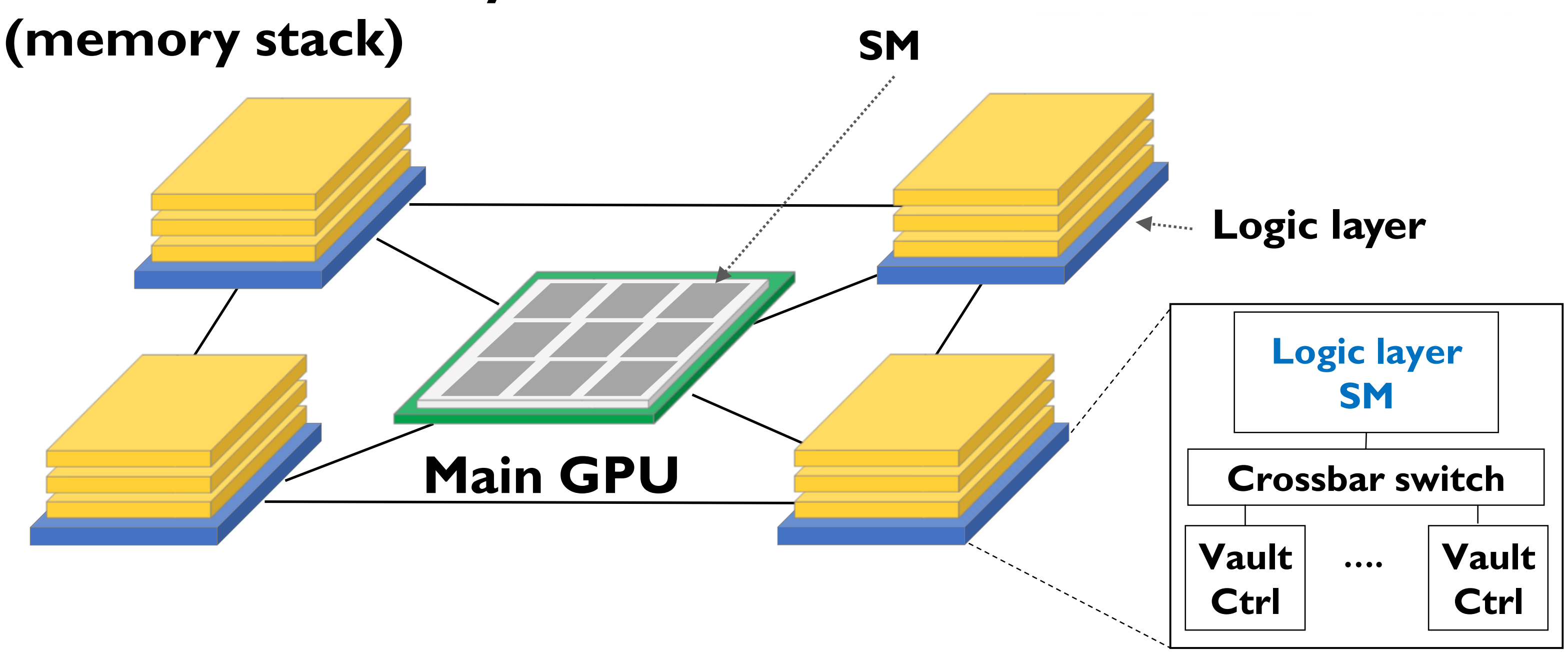


However, it requires significant programmer effort

Key Challenge 1

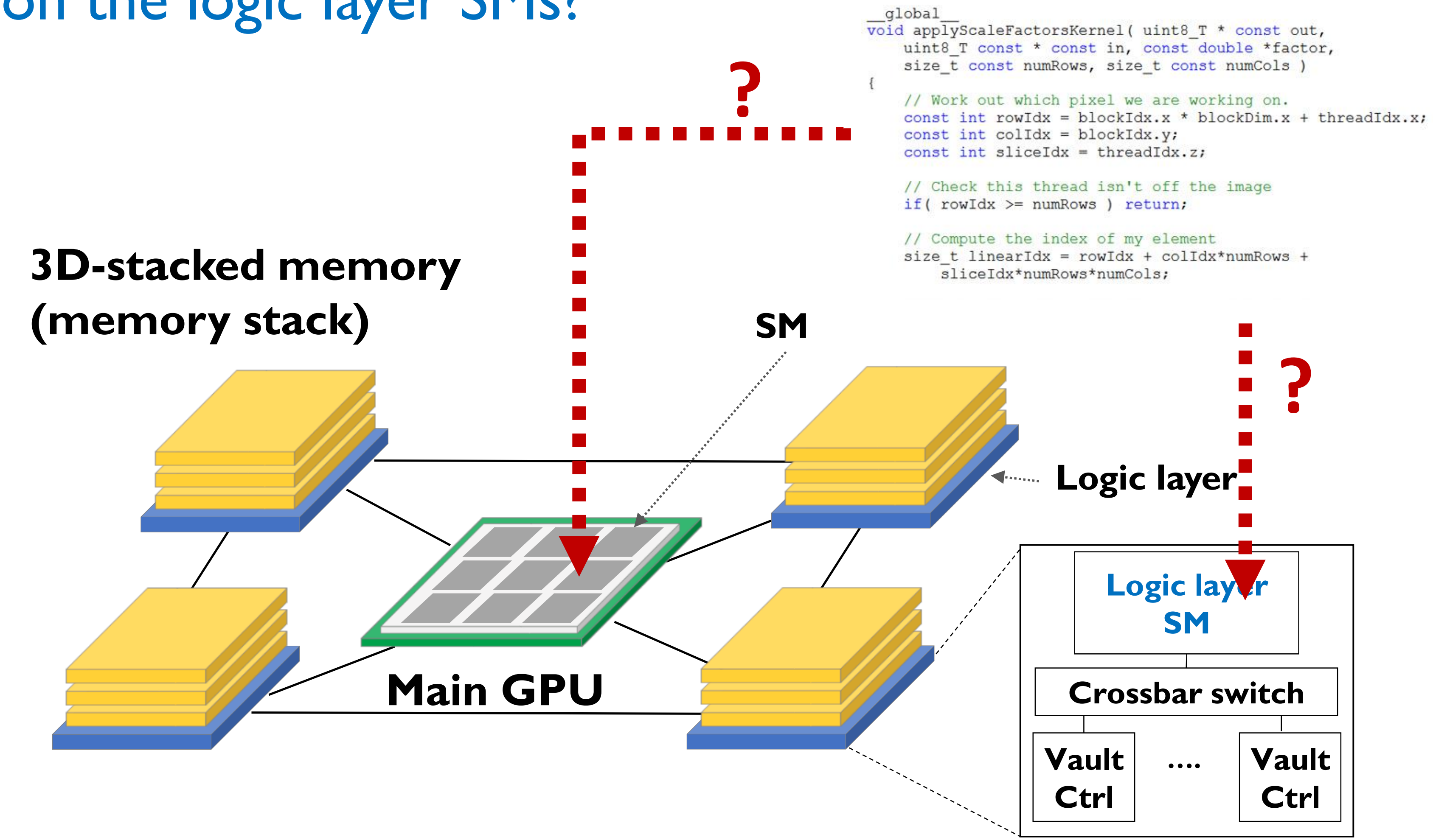


```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
    uint8_T const * const in, const double *factor,
    size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
        sliceIdx*numRows*numCols;
```



Key Challenge 1

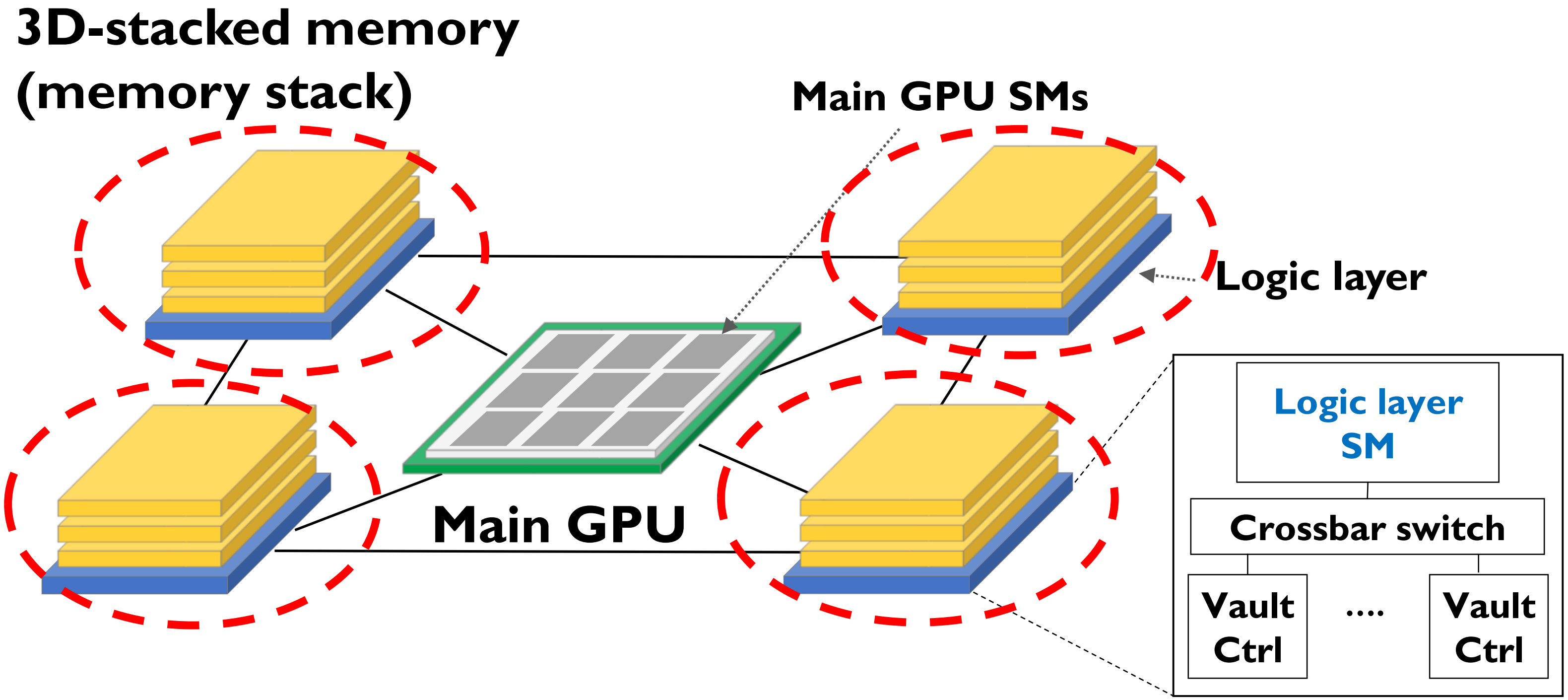
- **Challenge 1:** Which operations should be executed on the logic layer SMs?



```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
    uint8_T const * const in, const double *factor,
    size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
        sliceIdx*numRows*numCols;
```

Key Challenge 2

- **Challenge 2:** How should data be mapped to different 3D memory stacks?



Our Approach: TOM

- A new mechanism to identify and decide what code portions to offload.
    - The compiler identifies code portions to potentially offload based on memory profile.
    - The runtime system decides whether or not to offload each code portion based on runtime characteristics.

Our Approach: TOM

- A new mechanism to identify and decide what code portions to offload.
  - The compiler identifies code portions to potentially offload based on memory profile.
  - The runtime system decides whether or not to offload each code portion based on runtime characteristics.
- A new, simple, programmer-transparent data mapping mechanism to maximize code/data co-location.

Our Approach: TOM

- A new mechanism to identify and decide what code portions to offload.
  - The compiler identifies code portions to potentially offload based on memory profile.
  - The runtime system decides whether or not to offload each code portion based on runtime characteristics.
- A new, simple, programmer-transparent data mapping mechanism to maximize code/data co-location.
- **Key Results**: 30% average (76% max) performance improvement in GPU workloads.

Talk at Monday 2:50pm (Session 3B)

Transparent Offloading and Mapping (TOM)

Kevin Hsieh

Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee,
Mike O'Connor, Nandita Vijaykumar,
Onur Mutlu, Stephen W. Keckler

SAFARI Carnegie Mellon NVIDIA KAIST ETH zürich