Storage Mapping

Previously
- Tiling
- Spring Break

Today
- Note that Quiz 2 is due by Friday night and your intermediate reports are due next week
- Storage mapping for a 1D stencil computation
- Theory: what form? When legal?
- ScOP for smith waterman and Pluto
- Storage mapping possibilities for smith waterman
- Idea for LCPC paper based on semester projects

1D Stencil Computation Example

Example
// assume u[i] initialized to some values
for (s=1; s<T; s+=2) {
    for (i=1; i<(N-1); i++) {
        tmp[i] = 1/3 * (u[i-1] + u[i] + u[i+1]); // S1
    }
    for (j=1; j<(N-1); j++) {
    }
}
Analyzing the 1D stencil example

The iteration space …
  – For S1
  – For S2

How do we embed these in the same space?
  – Put j’s after i’s
  – Put j’s above i’s

How does the code change if it becomes single assignment?
  – Do array expansion for both u[] and tmp[]
  – Draw pictures of how the arrays are being accessed, notice that there are holes
  – How could we rewrite the code to use half of the storage?

Data reuse in the code

Two axes of data reuse
  – Temporal and/or spatial
  – Value and/or storage reuse

Data reuse within the original computation
  – One written value is put into one storage location and used by three computations
    – value and temporal
    – also spatial and storage because value is only stored in one location
  – Iteration i+1 reads a shifted set of values from iteration i
    – Value and temporal for the two reads for overlapping inputs
    – Spatial for the reads that are next to each other in memory
  – Iteration t+2,i overwrites the same two memory locations as iteration t,i
Ping-Pong compared with a mod mapping

Example (ping-pong)

```c
// assume u[i] initialized to some value
for (s=1; s<T; s+=2) {
    for (i=1; i<(N-1); i++) {
        tmp[i] = 1/3 * (u[i-1] + u[i] + u[i+1]); // S1
    }
    for (j=1; j<(N-1); j++) {
    }
}
```

Example (modular mapping) <show how to do this in AlphaZ>

```c
// assume u[0][i] initialized to some value
for (s=1; s<T; s++) {
    for (i=1; i<(N-1); i++) {
        u[(t+1)%2][i] = 1/3 * (u[t%2][i-1] + u[t%2][i] + u[t%2][i+1]);
    }
}
```

Storage mapping in general

Specify storage mappings with functions
- For example, write in iteration i,j should be stored in i
- Part of the storage mapping done by compilers is hidden from users
  - Row major
  - Column major

Occupancy vector indicates iterations that share storage
- ov = i-j, then iterations i and j share storage

Determining if a storage mapping is legal
- When a schedule is specified
- For any schedule, universal occupancy vector
Static Control Programs <show smith waterman with PLUTO>

Static Control Programs Definition
- Symbolic constants are variables that are not modified in the loop
- Loop bounds are affine combinations of loop variables and symbolic constants
- Array accesses are affine combinations of loop variables and symbolic constants
- If conditions are affine functions of loop variables and symbolic constants
- All function calls are pure
- In Pluto: change data if conditions to ternary expressions
- In Pluto: min and max are keywords! Do not use them as var names

References
- Pluto_general_notes.txt (see progress web page)
- Pluto/doc/DOC.txt
- Section 2.2 of the Feautrier paper

Storage mapping for smith waterman

What is the output of the computation?

Assume a computation like smith waterman …
- But only the last row of the 2D array is output
- What is a UOV for this data flow pattern?
- What is the corresponding data mapping?
- How can we implement a storage mapping with macros?
**LCPC Paper idea**

**Some LCPC history**

**Basic outline of the paper**
- Survey of the current polyhedral model power
- Case studies that evaluate the current tools based on…
  - Learning curve
  - Ease of use: documentation and robust error messages
  - Amount of tweaking needed to approach “best” possible performance
  - Resulting performance
  - Implementation limitations: what algorithms are not available
- Future research directions

**Student semester long projects**

**Case studies**
- Greg: LMie computes the scattering properties for polydisperse homogeneous spherical particles using the Mie solution, POCC
- Jared: Wavelets have become increasingly important in efficient video and image encoding. Pluto
- Glenn: genetic algorithm, POET
- Lixing: embedded applications benchmark, POET
- Matt: nearest neighbor algorithm for data mining, omega
- Brendan: libquantum Shor's algorithm for integer factorization, AlphaZ
- Nirmal: polyhedral benchmark suite, AlphaZ and PLUTO
- Ryan: support vector machine based learning algorithm, omega
- Steve: shortest path, AlphaZ
- Wenxiang: WalLS for solving SAT problems, POCC
Next Time

Lecture
- Transformation review
- Intro to algorithms needed for automation

Schedule
- Quiz 2 due March 23rd
- Project intermediate report due March 28th
- April 3rd will be a lab day during class, Manaf will help people with AlphaZ and Pluto. Distance students can email questions or use the discussion board.
- HW6 and HW7 will BOTH be due April 4th
for \( s = 1; \ s < T; \ s++ \) {
  for \( i = 1; \ i < (N-1); \ i++ \) {
    \( s1: \text{tmp}[i] = \frac{1}{3}(u[i-1] + \ldots) \)
  }
  for \( j = 1; \ j < (N-1); \ j++ \) {
    \( s2: u[j] = \frac{1}{3}(\text{tmp}[j-1]) \)
  }
}

**Iteration space S1**: \( \{[s, i] | 1 \leq s < T \land \exists a: 2a + 1 = s \land 1 \leq i \leq N - 2 \} \)

**Iteration space S2**:

\[ \text{embedding } \phi: \]

Embedding 1: \( \{[s, i] | \text{same } s \text{ constraints } \land \frac{1}{N-2} \leq i \leq \frac{N-2}{2N-4} \} \)

Embedding 2: \( \{[s, i] | 1 \leq s \leq T \land \exists a: 2a = s \land 1 \leq i \leq N - 2 \} \)
Single assignment

\[
\text{for } (s = 1; s \leq T; s++) \{ \\
\quad \text{if } (s-1)^2 == 0 \{ \\
\quad\quad \text{for } (i = 1; \\
\quad\quad\quad \text{tmp}[s, i] = \frac{1}{3} \ast (u[s-1, i]) \\
\quad\quad \} \\
\quad \} \\
\} \\
\] 

\[
\text{double } u[T+1][N], \text{tmp}[T+1][N]; \\
\]

\[
\text{for } (s = 1; s \leq T; s++) \{ \\
\quad \text{for } (i = 1; i < (N-1); i++) \{ \\
\quad\quad u[s, i] = \frac{1}{3} \ast (u[s-1, i-1]) \\
\quad \} \\
\} \\
\]

Storage $2(T+1)N$
multi-dimensional array mappings

\[ A[i][j] \]

double \[ A[N][M] \]

```c
#define A(i, j) i * M + j
```

\[ A[s_1][s_2]...[s_m] \]

**row major**

```c
#define A(i_1, i_2, ... i_m) i_1 * (s_2, s_m) + i_2 * (s_3, s_m)
```

**column major**

```c
#define A(i_1, i_2, ... i_m) i_m * (s_1, s_m) +
                         i_{m-1} * (s_1, s_{m-1}) +
                             ... +
                             i_1
```