Tile Communication Management

Louis-Noël Pouchet

pouchet@cse.ohio-state.edu

Dept. of Computer Science and Engineering, the Ohio State University

February 2012

888.11

Matrix-multiply (main)

Example

```
for (i = 0; i < M; ++i)
  for (j = 0; j < N; ++j)
     for (k = 0; k < P; ++k)
        C[i][j] += A[i][k] * B[k][j];
```

Matrix-multiply (tiled by 32)

Example

```
if ((M >= 1) && (N >= 1) && (P >= 1)) {
  for (c0=0;c0<=floord(M-1,32);c0++) {
    for (c1=0;c1<=floord(N-1,32);c1++) {
      for (c2=0;c2<=floord(P-1,32);c2++) {
        for (c3=32*c0;c3<=min(M-1,32*c0+31);c3++) {
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++) {
            for (c5=32*c2;c5<=min(P-1,32*c2+31);c5++) {
              C[c3][c4]+=A[c3][c5]*B[c5][c4];
            }
          }
        }
      }
    }
  }
}
```

Matrix-multiply (tiled by 32)

Example

```
if ((M >= 1) && (N >= 1) && (P >= 1)) {
  for (c0=0;c0<=floord(M-1,32);c0++) {
    for (c1=0;c1<=floord(N-1,32);c1++) {
      for (c2=0;c2<=floord(P-1,32);c2++) {
        // Start tile. Copy-in code should happen here.
        for (c3=32*c0;c3<=min(M-1,32*c0+31);c3++) {
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++) {
            for (c5=32*c2;c5<=min(P-1,32*c2+31);c5++) {
              C[c3][c4]+=A[c3][c5]*B[c5][c4];
            }
          }
        }
        // End tile. Copy-out code should happen here.
      }
    }
  }
}
```

Copy-in code

Key points:

- All data read in the tile must be fetched.
- This is the image, for each read array reference, of the tile iteration domain by the access function.
- **Copy happens to a local buffer**: we need to re-index.
- Buffer size needs to be computed.

Matrix-multiply (tiled by 32)

Example

```
if ((M >= 1) && (N >= 1) && (P >= 1)) {
  for (c0=0;c0<=floord(M-1,32);c0++) {
    for (c1=0;c1<=floord(N-1,32);c1++) {
      for (c2=0;c2<=floord(P-1,32);c2++) {
        // Copy-code for C:
        for (c3=32*c0;c3<=min(N-1,32*c0+31);c3++)
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++)
            C_local[c3-32*c0][c4-32*c1] = C[c3][c4];
        // Start tile.
        for (c3=32*c0;c3<=min(M-1,32*c0+31);c3++) {
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++) {
            for (c5=32*c2;c5<=min(P-1,32*c2+31);c5++) {
              C_local[c3-32*c0][c4-32*c1]+=A[c3][c5]*B[c5][c4];
            }
          }
        }
        // End tile. Copy-out code should happen here.
      }
    }
  }
}
```

Copy-out code

Key points:

- All data written in the tile must be stored back.
- This is the image, for each written array reference, of the tile iteration domain by the access function.
- **Copy from a local buffer**: we need to re-index.

Matrix-multiply (tiled by 32)

Example

```
if ((M >= 1) && (N >= 1) && (P >= 1)) {
  for (c0=0;c0<=floord(M-1,32);c0++) {
    for (c1=0;c1<=floord(N-1,32);c1++) {
      for (c2=0;c2<=floord(P-1,32);c2++) {
        // Copy-code for C:
        for (c3=32*c0;c3<=min(N-1,32*c0+31);c3++)
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++)
            C_local[c3-32*c0][c4-32*c1] = C[c3][c4];
        // Start tile.
        for (c3=32*c0;c3<=min(M-1,32*c0+31);c3++) {
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++) {
            for (c5=32*c2;c5<=min(P-1,32*c2+31);c5++) {
              C_local[c3-32*c0][c4-32*c1]+=A[c3][c5]*B[c5][c4];
            }
          }
        }
        // End tile.
        // Copy-code for C:
        for (c3=32*c0;c3<=min(N-1,32*c0+31);c3++)
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++)
            C[c3][c4] = C_local[c3-32*c0][c4-32*c1];
      }
    }
  }
}
```

Step-by-step

1. Tile the original code
   - Hint: may be useful to insert the copy statements before tiling
2. Insert copy-in code
   - Hint: re-indexing can be done syntactically here.
   - Hint: place it right before the tile at first.
3. Insert copy-out code
   - Hint: place it right after the tile at first
4. Compute the buffer size
   - Hint: this is the max of the number of data points communicated
5. Optimize communications

Communication Optimization

- Communications may be **hoisted** to outer tile loops
  - Criterion: the communication is invariant for the tile loop
  - Must hold for both copy-in and copy-out
- Communications may be pipelined (double-buffer)
  - Method: use two buffers, and software-pipeline
- What about inter-tile reuse?

Matrix-multiply (tiled by 32)

Example

```
if ((M >= 1) && (N >= 1) && (P >= 1)) {
  for (c0=0;c0<=floord(M-1,32);c0++) {
    for (c1=0;c1<=floord(N-1,32);c1++) {
      // Copy-code for C:
      for (c3=32*c0;c3<=min(N-1,32*c0+31);c3++)
        for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++)
          C_local[c3-32*c0][c4-32*c1] = C[c3][c4];
      for (c2=0;c2<=floord(P-1,32);c2++) {
        // Start tile.
        for (c3=32*c0;c3<=min(M-1,32*c0+31);c3++) {
          for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++) {
            for (c5=32*c2;c5<=min(P-1,32*c2+31);c5++) {
              C_local[c3-32*c0][c4-32*c1]+=A[c3][c5]*B[c5][c4];
            }
          }
        }
        // End tile.
      }
      // Copy-code for C:
      for (c3=32*c0;c3<=min(N-1,32*c0+31);c3++)
        for (c4=32*c1;c4<=min(N-1,32*c1+31);c4++)
          C[c3][c4] = C_local[c3-32*c0][c4-32*c1];
    }
  }
}
```