Compiler-based automatic data prefetching with hardware support

Seungryul Choi, Gautham K.Dorai / Donald Yeung

Motivation for Data Prefetching

- CPU clock speed doubles every 18 months
- Memory speed increases 7% every 18 months
- Cache miss penalty increases exponentially

Real World Applications

- Numeric Applications: Easy to prefetch
  - Scientific codes

```
for (i = 1 ; i < 1000 ; i++) {
    B[i] = A[i] + B[i - 1];                linear array access
    .....
}
```

- Non-numeric Applications: Hard to prefetch
  - Artificial Intelligence, Database, Compiler and Graphic applictions

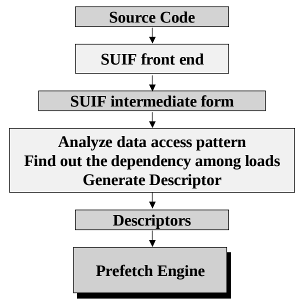
```
for (i = 0 ; i < 5 ; i++) {                        short loop
    for (Y = X[i] ; Y != NULL ; Y = Y ->next) {    unknown length
        foo(Y->data);                              inter-procedure analysis
        if (Y->tag == NULL)                        unknown amount of work
            break;                                 conditional exit
        else
            sum += Y->size;                        conditional data access
        .....
    }
}
```

Why Compiler and Hardware Cooperation

- Pure Software Prefetching Schemes
  - (+) Compiler can extract high level information from source code
  - (-) More run time overhead & More storage
- Pure Hardware Prefetching Schemes
  - (+) Low runtime overhead
  - (-) Less accuracy & Less coverage
- Prefetch Engine with Compiler Support
  - **Benefits of both schemes**

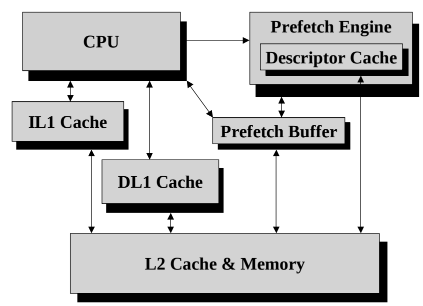
Compiler Support

- A SUIF path generates **Descriptor**: high level description of data structure
- The Descriptor provides sufficient information to regenerate all the addresses of the data structure
- The Descriptor is general enough to express very complex data structures
  - array, linked list and combination of array and linked list



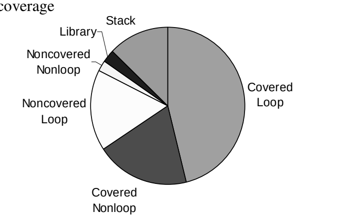
Hardware Support

- Prefetch Engine: generates address for prefetching and issues prefetches
- Prefetch Buffer: stores prefetched data
- Descriptor Cache: 1Kbyte size - stores 64 descriptors

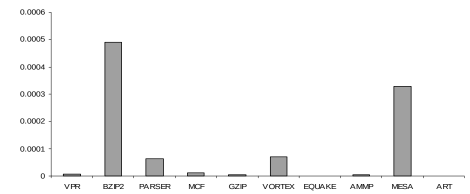


Observations

- Test programs: 6 SPEC CPU INT2000 and 4 SPEC CPU FP2000
- Cache miss coverage



- Descriptor Cache miss overhead: 100~200 CPU cycles
- Descriptor Cache miss percentage (%)



Conclusions and Future Works

- Promising preliminary results from real world applications
- Good cache miss coverage due to
  - Very rich compile time information on data access pattern
  - Intelligent compiler makes this information available to prefetch engine
- Low run time overhead due to
  - Dedicated hardware for prefetching
  - Small Descriptor Cache size and very low miss rate
- Future works
  - Evaluate the overall performance of our “compiler-based prefetching with hardware support”
  - Compare the performance of our system with other prefetching schemes