# Malleable Architectures for Adaptive Computing
# MIT9904-04
# Arvind, Larry Rudolph and Srinivas Devadas

Our research over the past two years has focussed on malleable caches and processors for data-intensive computing. We have worked closely with a researcher at NTT on improving the memory and caching needs of speech applications. Hiroshi Sawada, our NTT colleague, visited us last year for about six months, and reported on collaborative work at the NTT Japan meeting in January. Hiroshi is now examining more sophisticated speech processing techniques and we feel that proper memory hierarchy management is even more important. In particular, the NTT team that Hiroshi is a part of is trying to separate different conversations from the same audio stream. Such an application is a combination of stream processing with complex table-lookup actions. It is precisely the combination of these two types of memory access patterns, that we found to be problematic on current microprocessors in our previous research. We developed cache control instructions to solve this problem in the past year (briefly described below), and believe that our techniques will enable such real-time speech processing. For applications that are memory intensive, a memory-centric approach to processing is needed.

We are sending an MIT graduate student from our group, Josh Jacobs, to visit NTT this summer. This will strengthen our already strong collaboration. In the forthcoming year, we propose to extend our results from the previous 2 years in the following significant ways.

**Cache control instructions for streaming applications:**

In order to significantly improve the performance of streaming applications, we have proposed the use of kill and keep cache control instructions. Using these instructions allows us to change the cache replacement policy for the better. We have proven theorems that provide conditions under which the introduction of these instructions is guaranteed to improve the hit rate. We will now use the theorems to derive algorithms that will automatically introduce these instructions into software, using lifetime analysis during compilation. These algorithms will automatically detect streaming data in applications.

**Cache Oblivious Transformations:**

We have proven theorems that show that if we can reorder a program's memory reference stream such that the reordered memory reference stream satisfies a disjointness property, then the transformed program corresponding to the reordered stream is guaranteed to have fewer misses for any cache with arbitrary size or organization so long as the cache uses the LRU replacement policy. We can now apply these results to reorder instructions within a basic block, to transform loops, to reorder blocks within a procedure, or to reorder procedures within a program so as to improve hit rate for any cache that uses LRU replacement. Based on these theorems, we will develop algorithmic methods for program transformation to improve cache performance.

**Memory-Centric Scheduling:**

We have proposed analytical cache models for time-shared systems, which estimate the overall cache miss-rate from the isolated miss-rate curve of each process when multiple processes share a cache. Unlike previous models, our model works for any cache size and any time quantum. The model has been used to allocate cache space to each process. In our examples, the model-based partitioning has improved the cache miss-rate up to 40% over the normal LRU replacement policy. We will now use the model to allocate jobs to each processor in multi-processor systems so as to minimize the total number of cache misses. The operating system will automatically profile the characteristics of each process, evaluate possible allocations, and change the process allocation.

**Collaboration**

There is one collaborative paper in progress.
Hiroshi Sawada came to visit LCS for 6 months
Larry Rudolph visited NTT
Josh Jacobs, an MIT graduate student will be visiting NTT this summer.