ALI and PLASMA Laboratories: CS354
Architecture and Language Implementation
Programming Languages and Systems at Mass.

Principal Investigators

Prof. Eliot Moss, CS372, moss@cs.umass.edu

Prof. Chas. (Chip) Weems, CS342, weems@cs.umass.edu

Prof. Emery Berger, CS344, emery@cs.umass.edu

Graduate Students

John Cavazos, CS354	Naren Sachindran, CS354
Yi (Eric) Feng, CS354	Pritesh Sharma, CS354
Matthew Hertz, CS354	Mathew Thoennes, IBM
Asjad Khan, CS354	Ed Walters, CS354
Tyler Maxwell, CS354	Zhenlin Wang, CS354
Trek Palmer, CS354	Ting Yang, CS354

Tim Richards, CS354

Research Programmer	Undergraduate Students
Chris Hoffmann, CS354	Adam Bellmore, Jim Dinan

Current Projects

- Understanding, improving, and exploiting data locality
- Dynamic adaptive optimization of Java
- Garbage collection algorithms and performance
- CRAMM: Cooperative Robust Automatic Mem. Mgmt.
- Scheduler-aware virtual memory management
- Micro-architectural features that exploit compile-time analyses at run time
- Co-Generating simulators and compiler back-ends from description files: CoGenT
- Cache architecture for high performance and embedded systems, e.g., DASAT
- Implementing Persistent and Distributed Java
- Applying machine learning in system optimization

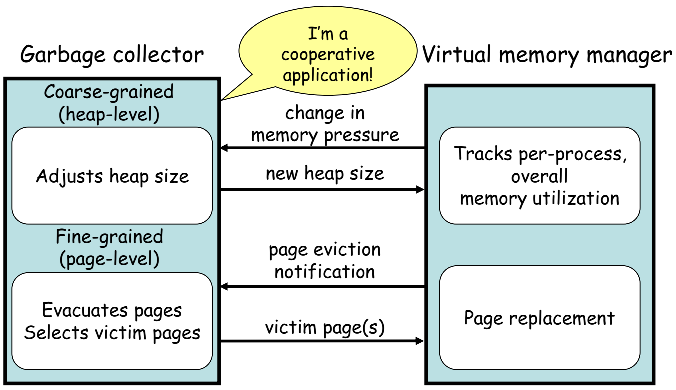
Laboratory Description: We work across the boundaries of computer architecture, language implementation, and operating systems. We believe that hardware can better inform compilers and run time systems of run time behavior, and that more controllable hardware, particularly caches and similar mechanisms, can be exploited by compilers and the run time. Likewise we believe that cooperation across the application-OS boundary can provide great leverage, for example between garbage collectors and virtual memory managers (VMMs) to improve paging of GCed applications (CRAMM project), and between OS schedulers and VMMs (SAVMM project).

We are particularly interested in improving memory performance for object oriented programming languages, especially Java. We use state-of-the-art Java virtual machines, including dynamic optimizing compilers, advanced garbage collection techniques, etc. We also employ hardware and OS simulators to explore performance in more detail, and the CoGenT project aims to make such exploration much easier through automatic generation of simulators and compiler components. Novel caches are an example of the hardware enhancements we aim to evaluate. Our results have been included in commercial Java Virtual Machines, and our work regularly appears in top systems conferences.

Other distinctive efforts include applying machine learning in compiler and run-time system tuning, such as instruction scheduling and garbage collector policy selection.

We are funded by several NSF grants, and hope to hire at least one new research assistant this year. A number of our graduates are faculty members at well-known institutions.

CRAMM Model of GC / VMM Interaction



The CoGenT prototype