Chip Multiprocessor
Parallel processors and systems-on-a-chip are becoming the norm.

Chip Multithreaded
Long memory latencies can be hidden by scheduling other hardware contexts.

Non-Uniform Memory Access
Local memories are inherently faster and lower power. Non-uniformity exposes the need for optimizing memory accesses.

Transactional Memory
Hardware support for nested transactions and rollback to support lock-free parallel execution.

Object-Oriented Memory
Addressing to support objects, legacy support, non-local memories and transactions. TLB caches data keeping memory accesses cheap.

Parallelization
Supercompiler
Supercompilers typically target mathematical languages extracting parallelism. We use the same approaches to dynamically create threads that can take advantage of lightweight threads and idle contexts.

Speculation
Use transactions to create work that can be discarded if a loop breaks out early, or if two pieces of code interfere through side-effects.

Locally Distributed VMs
Present the programmer with a single virtual machine, internally objects are distributed around memories, threads migrate to be close to their data and supplies of idle contexts.

Operating Systems
Integrating the VM and OS exposes optimisations across library and OS barriers. Having a “safe” language allows memory management to be simpler and faster. Scheduling improvements are exposed in both the VM and OS.

Legacy Support
Having a new architecture requires support if users are to migrate to it. FXI32 and Rosetta demonstrate the ability to migrate using dynamic binary translation. PearColator advances the approaches of FXI32 and Rosetta by utilizing the compiler performance of a VM optimizing compiler.

Hardware support for emulated virtual memory allows pages to be emulated as objects at consecutive addresses.