Affinity-aware Dynamic Pinning Scheduling for Virtual Machines

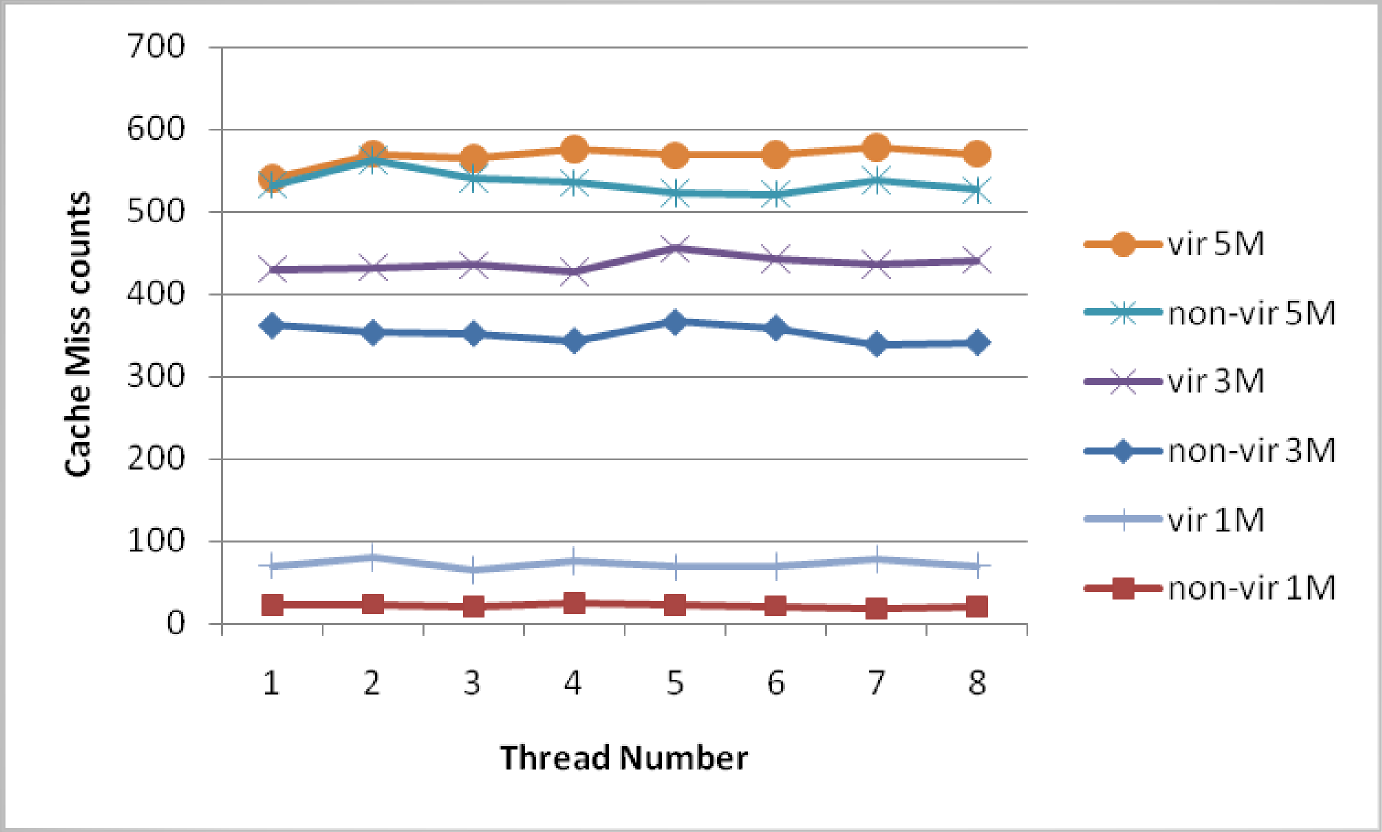
Zhi Li
lizhi@cse.buaa.edu.cn
School of Computer Science, BeiHang University,
Beijing, China

Outline

- Motivation
- CPU Affinity-aware Method
- Dynamic Pinning Scheduling
- Performance Evaluation

Motivation

L2 Cache misses on both non-virtualization and virtualization



Virtualization leads to worse cache miss

Analysis of the Issue

- VCPUs from same Domain take turns to run in a same CPU runq
- Frequent migrations from this kind of VCPU

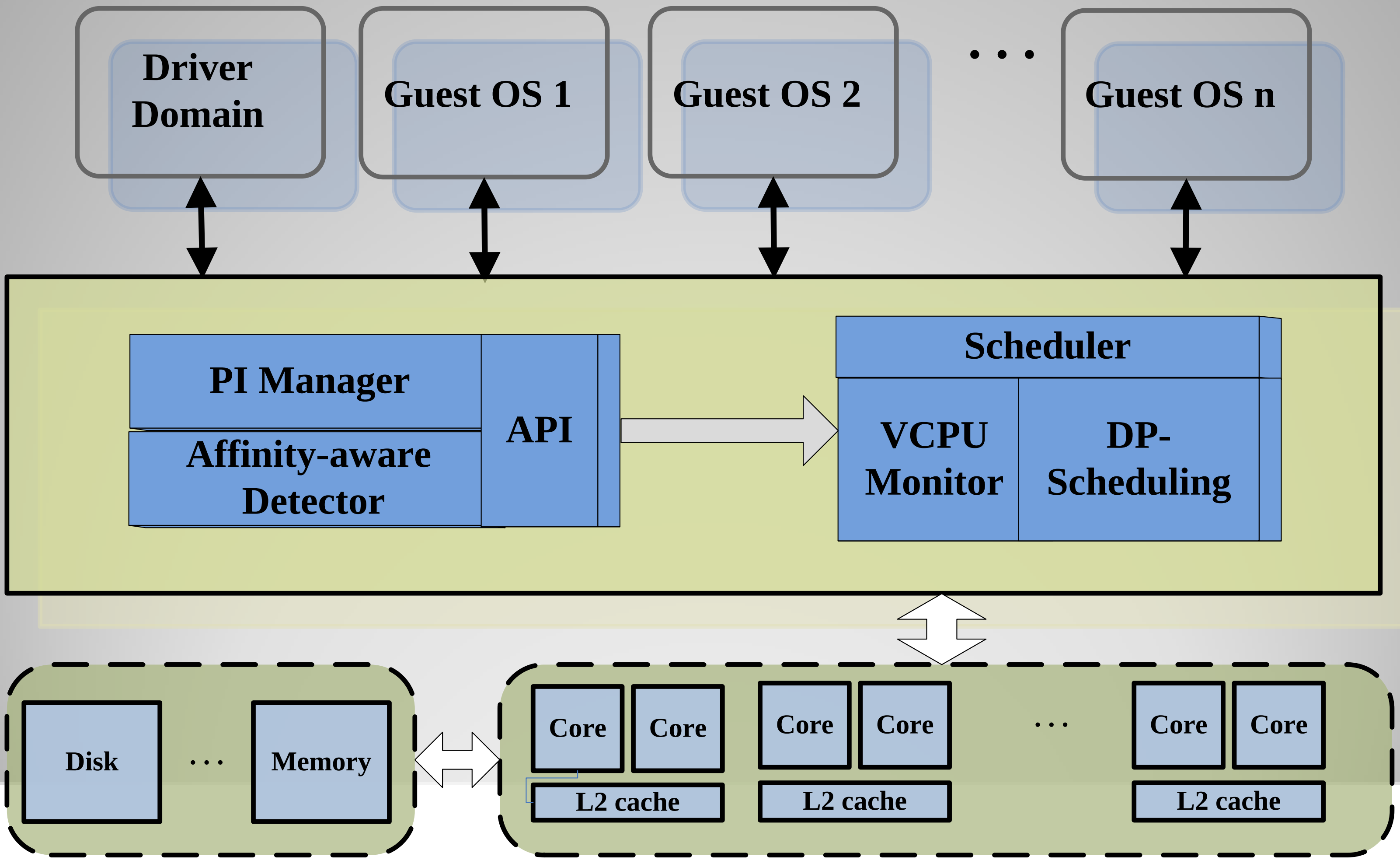
How to bridge the semantic gap between Guest OS and VMM

- Affinity-aware DP-Scheduling:
  - Affinity-aware method: providing the task affinity information to VMM.
  - DP-Scheduling: implementing that VCPU can be pinned or unpinned dynamically

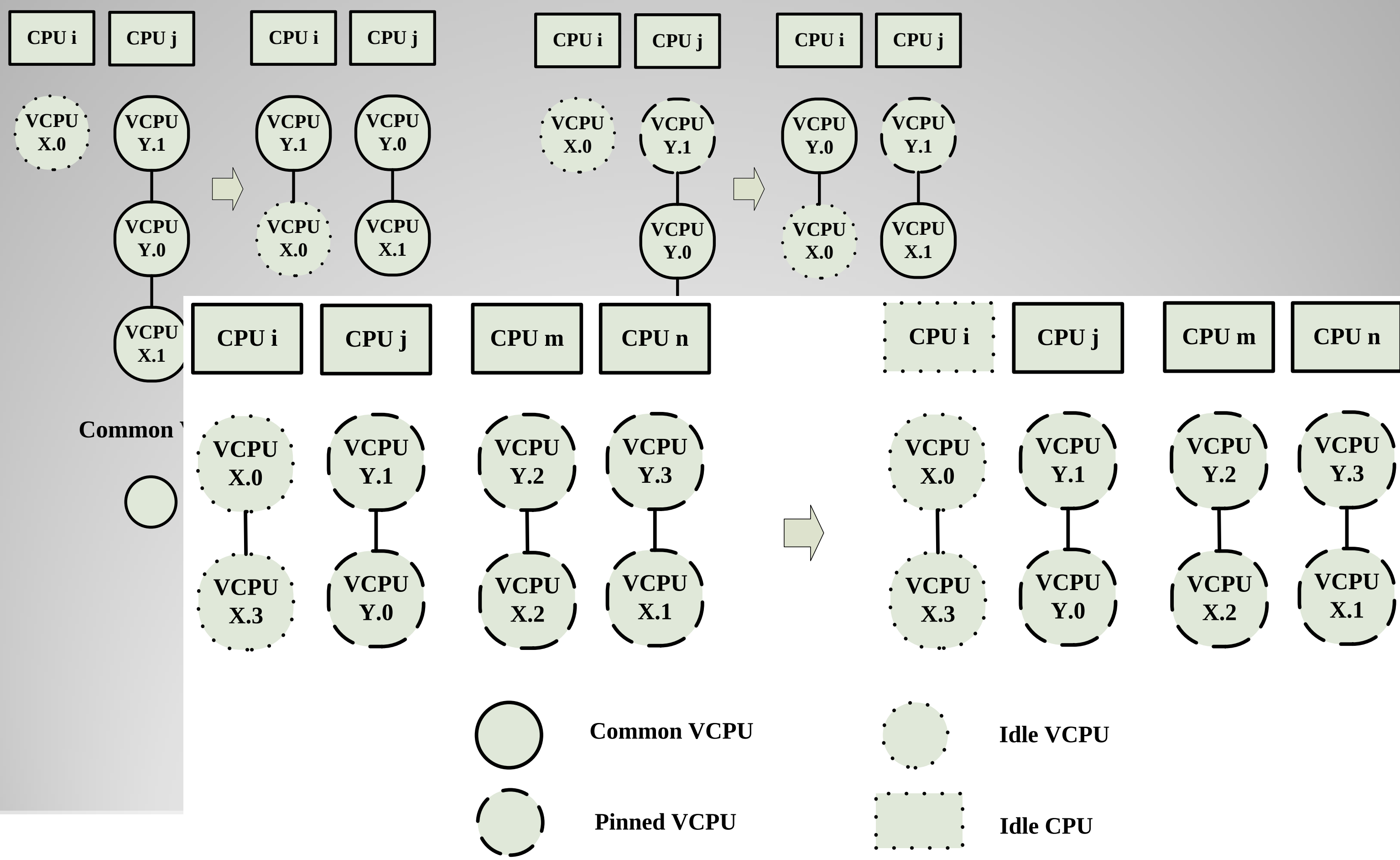
CPU Affinity-aware Method

- Timing Control
  - When CR3 is changing
- Methodology for Capture
  - Affinity Coefficient (AC)
- API
  - Provide AC for the scheduler

Dynamic Pinning Scheduling



Dynamic Pinning Scheduling



Dynamic Pinning Scheduling

- strategies :
  - (a).pin VCPU to the CPU with no pinned VCPU at this time.
  - (b).pin VCPU to the CPU with lower workload.
  - (c).pin VCPU to the local CPU if both (a) and (b) do not happen.
  - (d).do not migrate the VCPU actively when it is unpinned.
  - (e).unpin the VCPU with the lowest value of AC when the number of CPU equals.
  - (f).unpin the VCPU pinned before if it goes to the state of OVER or IDLE.

DP-Scheduling Algorithm

Let Φ stand for VCPU set from all domains, that is, $\Phi = \{VCPU_1, VCPU_2, \dots, VCPU_{|\Phi|}\}$. Let φ_p be the set of VCPUs that were pinned, whereas φ_{up} is the set of unpinning VCPUs. Hence, $\Phi = \varphi_p \cup \varphi_{up}$, and $|\Phi| = |\varphi_p| + |\varphi_{up}|$. Besides, let $\Upsilon = \{\gamma_1, \gamma_2, \dots, \gamma_{|\Phi|}\}$, where γ_i is the value of AC for $VCPU_i$.

```
For each VCPUi ∈ Φ {
    CPUx = VCPUi ->processor;
    /*VCPUi has been pinned by its high value of AC*/
    If (γi ≥ THRESHOLD and VCPUi ∈ φp) Return;
    /*VCPUi with its high value of AC was not pinned */
    Else If (γi ≥ THRESHOLD and VCPUi ∈ φup
    and VCPUi ->pri > OVER){
    peer_cpu = select_peer(VCPUi);
    set_unmigrateable(peer_cpu, vcpu);
    del(φup, VCPUi);
    add(φp, VCPUi);
    }
    /*Pinned VCPUi contains low value of AC currently
    *according to strategy (d)*/
    Else If (γi < THRESHOLD and VCPUi ∈ φp){
    set_migrateable(CPUx, VCPUi);
    del(φp, VCPUi);
    add(φup, VCPUi);
    }
    /*The total number of VCPUs should not over
    *CPU number -1*/
    If (|φp| == |CPU|)
    set_migrateable_by_ac(φp, φup);
}
```

Performance Evaluation

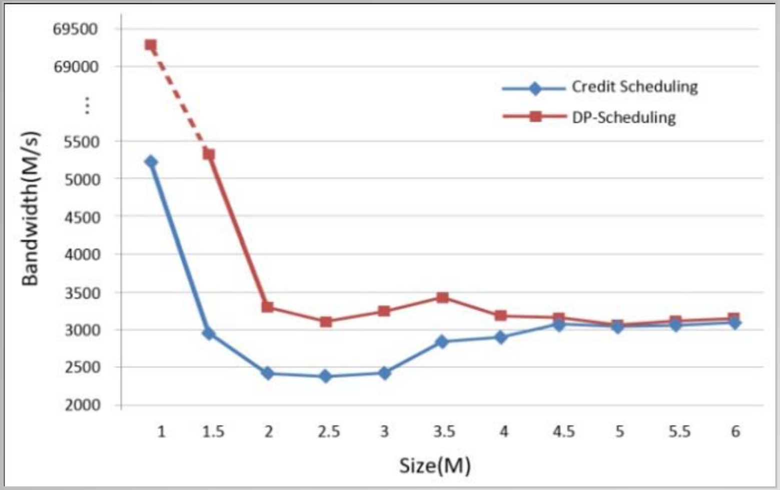
Platform

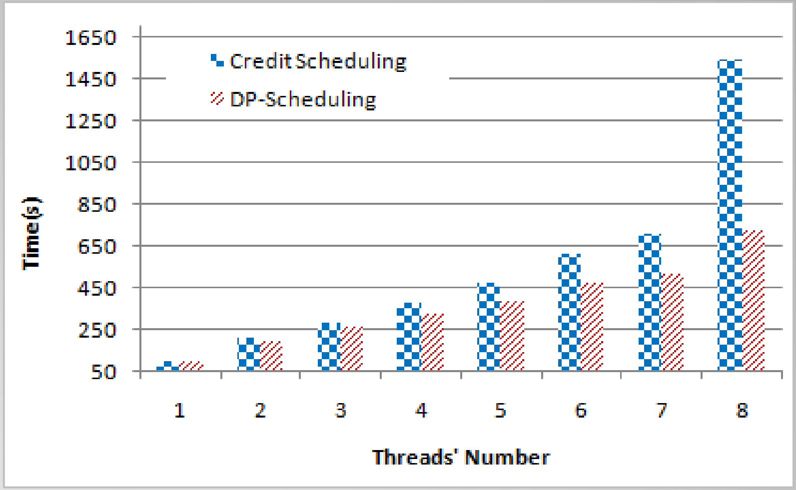
- Xeon 5405(two quad-core)
- L2 cache:6M
- RAM:4G
- Xen:3.4.3

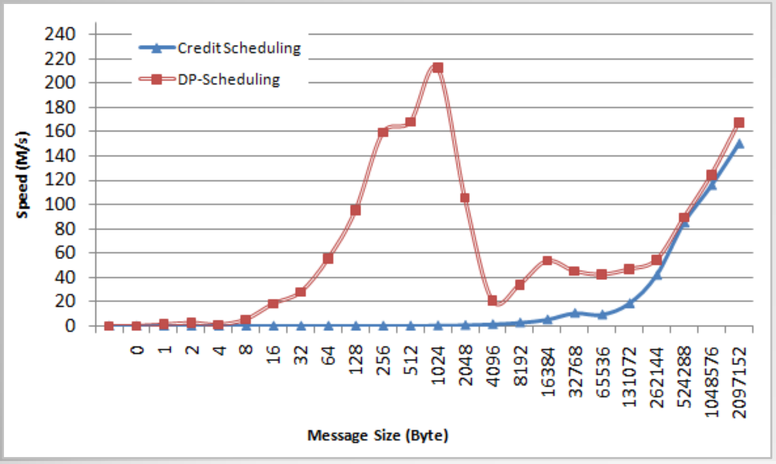
Benchmark

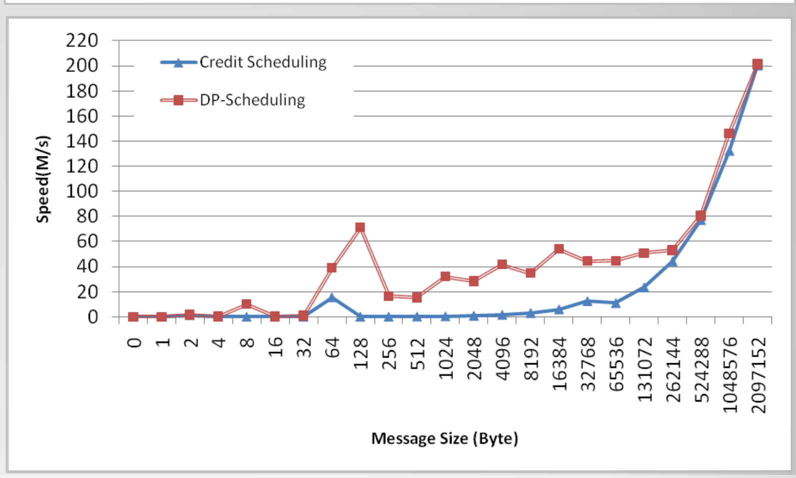
Benchmark Category	Code Name	Variable	Measurement
HPCC	STREAM	Array size	Memory Bandwidth
EPCC	OpenMP Micro-benchmark suite	Thread Number	Time
IMB	Sendrecv Exchange	Message Size	Transfer Speed

Performance Evaluation









- Conclusion

DP-Scheduling outperforms Credit scheduling for kinds of CPU-bound tasks, without interfering the load balance

Thank You !