CSC 714 Project Progress Report 1

Analyzing the Effect of Predictability of Memory References on WCET

Amir Bahmani(abahman@ncsu.edu), Vishwanathan Chandru(vchandr6@ncsu.edu)

Objective:

To develop a colored malloc which colors MC and allocates memory according the colors based on various parameters like proximity to the tile, size of allocation, network contention etc.

Task Status:

1) **Ramp up on Tilera architecture and APIs**

Owner(s) : Both

Targeted completion : 24th March 2014

Status : Completed

Description : In this task, we were supposed to go through the Tilera architecture and APIs. We were successfully able to run the applications with the NoC library but for some reason couldn't run the applications without the shepherd running. Hence we will stick to using NoC library. We were successfully able to configure the accounts.

Open actions: PCI examples not running. Need to figure out why they are not working but its **low priority**.

2) **Figuring out base parameters for colored malloc**

Owner(s) : Vishwanathan

Targeted Completion: 24th March 2014

Status : Completed

Description : I started out looking into it to figure out what parameters could consider for color based allocation. I did some basic investigation to understand and figure out what can and cannot be done. I went through the architecture documentations of Tilera and hypervisor internals. The focus was on two key parameters proximity and striping. For proximity, I found that there are calls to figure out the closest MC. Hence decided we can consider proximity as one of the factors. The second factor is how we strip the address range. There were two options, either we do it based on virtual address or we do it on basis of physical address. I decided to go for virtual address basis as it will contiguous and will hide the effects of paging as the paging causes the physical address may change continuously and may result into clogging of a MC. The striping will be based on size of allocation in question. If it is small we could go for colouring in an interleaved fashion, if it is a huge allocation than we do it in a block interleaved fashion so as to ensure minimum contention and parallelization of memory access. We plan to try both the approaches with varying allocations and observer varying latencies.

Open Actions: None regarding implementation as the goal was just to analyze the feasibility of parameters

3) **Figure out way to remap the memory accesses**

Owner(s) : Vishwanathan

Targeted Completion: 31st March 2014

Status : In progress(very high probability of a slip)

Description : The aim of this task was to figure out dynamic remapping of memory accesses. I started off with the investigating the possibility of intercepting the L2 misses. The structures governing client configurations were identified namely client_config, but could not locate any library static or dynamic which exports the symbols for the necessary functionality. The already exported symbols available were analyzed and were not of any use. It was concluded that to carry on in this direction it would require a hypervisor rebuild and deployment. Then as per the suggestion from Professor, this was investigated from a different angle. Instead of remapping on the fly, if we could do the loading of pages into memory adhering to the stripping and controller assignment done at virtual address level, the task will become much easier and effective. During the investigation it was found that hypervisor is called only when needed and one particular need which could solve problem of mapping is page fault handling. It does something called ***downcall***, which is basically telling the underlying OS to act if we need kernel to act. In our context, if the page couldn't be located on L1 or L2 cache(s) than control is transferred to OS to handle the PF. We can take advantage of this downcall to handle the PF the way we want and this allows us to control both user and kernel space allocations. This approach is similar to the approach taken in "PALLOC: bank aware memory allocator". Also as part of the investigation, it was found that the 36 bit address used in MDN packets can be used to identify the bank, row, MC Etc. Three types of addresses were identified 32 bit VA, 64 bit client physical address as seen by hypervisor, 64 bit PA as seen by MC.

Open Problems:

1) Page table functions are identified and corresponding interrupt handlers are located. Need to understand the flow to tap onto those functions and modify page loading.

2) Need to figure out how to build and deploy the modified Linux libraries. I could locate an android 3.0 build and a master's project involving porting Barrel Fish onto Tilera. Interestingly these involve downcall modifications. Given that and source code for hypervisor we could define a design and implementation for colored malloc.

3) Need to understand the exact control flow once the control enters the Linux kernel so as to modify allocations. If needed get in touch with the following people:

-> People in the Barrel Fish port

-> Try reaching out to Tilera or some forum to understand the downcall handling for page faults.

4) **Figure out a way to figure out the corresponding MC from physical address**

Owner(s) : Amir

Target Completion: 31st March 2014

Status : In Progress

Description : As a part of this task, it was required to identify the translation from VA to PA and from PA decode which MC to assign to. The translation mechanism using page map and page tables were understood and given the fact that downcall mechanism is used for PF handling, there is a strong chance that same might work for Tilera board.

Open Actions :

1) There are 3 addresses, CPA, PA and VA, looks like what we get is physical address but there are calls for querying page table and associated things. This needs to be sorted out before the implementation of malloc could be done.
2) Current flow regarding address translation is not well understood. This understanding is needed for implementation for malloc. I might have to try reaching out to Tilera via discussion forum to understand.

5) **Memory mapping strategy**

Owner(s) : Amir

Target Completion: 7th April 2014

Status : To Do

Description : As a part of this task, it is planned to determine an algorithm for mapping of memory to MC. It will be done based on the parameters already determined in the previous tasks i.e. proximity and size of requested memory.

6) **Strategy for contention analysis**

Owner(s) : Vishwanathan

Target Completion: 7th April 2014

Status : To Do

Description : As a part of this, I plan to measure the impact of NoC contention using repeated memory accesses considering only RAW, WAR and WAW dependency. This has to be done to ensure that there is no caching in the L1 and L2 cache. One malloc is implemented it will help us analyze the impact of memory access distribution among MCs.

7) **Malloc implementation**

Owner(s) : Both

Target Completion: 14th April 2014

Status : To Do

Description : As part of this task, we will work together and implement the malloc.

8) **Test Cases Design and Implementation**

Owner(s) : Both

Target Completion: 14th April 2014

Status : To Do

Description : Test case application for the purpose of contention analysis and effectivity of colored malloc.

9) **Task Mapping Algorithm and implementation**

Owner(s) : Amir

Target Completion: 21st April 2014

Status : To Do

Description : This task is part of our secondary objective under which we investigate optimal placement of inter dependent task. It will be attempted once we are done with primary objective.

10) **Task Mapping algorithm validation**

Owner(s) : Vishwanathan

Target Completion: 25th April 2014

Status : To Do

Description : Test development and validation of task mapping

11) **Presentation**

Owner(s) : Amir

Target Completion: 27th April 2014

Status : To Do

Description : Demo of the malloc prototype and task mapping(if task mapping done).

12) **Final Report**

Owner(s) : Both

Target Completion: 1st May 2014

Status : To Do

Description : Final report submission

References:

[1] Hyoseung Kim , Dionisio de Niz, Bj¨ orn Andersson† , Mark Klein†, Onur Mutlu, Ragunathan (Raj) Rajkumar , Bounding Memory Interference Delay in COTS-based Multi-Core Systems in COTS-based Multi-Core Systems

[2] Heechul Yun , Renato Mancuso , Zheng-Pei Wu , Rodolfo Pellizzoni, PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms

[3] Borislav Nikoli´ c, Patrick Meumeu Yomsi and Stefan M. Petters, Worst-Case Memory Traffic Analysis for Many-Cores using a Limited Migrative Model

[4] https://android.googlesource.com/kernel/common.git/+/android-3.0/

[5] Tilera Architecture documentation

[6] Porting Barrelfish to the Tilera TILEPro64 Architecture, ROBERT RADKIEWICZ and XIAOWEN WANG, KTH Information and Communication technology