Design of Memory Allocation Algorithm Based on Forecast-union Principle in Heterogeneous Multi-core Architecture

Xiaohui Cheng, Congcong Xia and Jinzhou He

ABSTRACT

Analyzing the problems of timeliness and fragment rate from the perspective of timing and space in heterogeneous multi-core architecture, this paper proposes the forecast-union mechanism based on forecast model. From the view of timing, predicting the required memory size based on the history information of memory application, and allocating memory block in advance to reduce the time of waiting for the system call; From the view of space, in order to reduce the fragment rate by reducing the frequency of cutting blocks apart, allocating a big block consisted of the required block and the allocated block to the process. The contrastive experiment result shows that the forecast-union allocation algorithm is efficient in solving the problem of timeliness and fragment rate.

KEYWORD: Heterogeneous multi-core; Timeliness; Fragment rate; Forecast-union allocation

INTRODUCTION

The memory space is a limited but indispensable resource in a computer system, and the system performance is related with the appropriate management and usage of memory space. Therefore, a qualified memory management strategy is critical for improving the overall system performance. This paper designs a forecast-union allocation mechanism based on the process history information from the perspective of timeliness and memory fragment rate. On the one hand, to reduce
the allocating time, the basic principle of the forecast-union mechanism is allocating the required memory in advance by a corresponding prediction thread; on the other hand, to reduce the fragment rate, allocating a coalescent memory consisted of the current required memory and the predicted memory by using the principle of union allocation.

**PREDICTION THEORY MECHANISM**

The Monte Carlo chain is a discrete time stochastic process that has the Monte Carlo property. In the case of given information, the past conditions are not related with the prediction of the future during the stochastic process. The Monte Carlo process is a special process which has no aftereffects, the "future" situation is independent from the "past" situation, and is only related with the "current" situation.

Definition: Assuming the state space of \( \{X(t), t \in T\} \) is \( S \), if \( n \geq 2 \) and \( t_1 < t_2 \cdots < t_n \in T \), the conditional distribution function of \( X(t_n) \) equals to the conditional distribution function when \( X(t_{n-1}) = x_{n-1} \) under the condition that, that is:

\[
P\{X(t_n) \leq x_n | X(t_1) = x_1, X(t_2) = x_2, \cdots, X(t_{n-1}) = x_{n-1}\} = P\{X(t_n) \leq x_n | X(t_{n-1}) = x_{n-1}\}, x_n \in R
\]

then \( \{X(t), t \in T\} \) is a Monte Carlo process.

According to the Monte Carlo process, the system state is \( i \) when \( t = n \), then the state transition probability is \( P\{X(n+m) = j | X(n) = i\} \), which is also \( p_{ij}(n, m+n) \) or \( p_{ij}^{(m)}(n) \), after \( m \) steps to transitions to the state \( j \). When \( m = 1 \), the state transition probability is also called the Monte Carlo transition probability.

Before the Monte Carlo prediction, the states of a process have to be divided into \( k \) separate states, the transition probability between these states is \( p_{ij} \), which is the probability of the transition from state \( i \) to state \( j \), all the transition probabilities of these states can be organized into a \( K \times K \) state transition probability matrix \( P \):

\[
P = \begin{bmatrix}
p_{11} & p_{12} & \cdots & p_{1k} \\
p_{21} & p_{22} & \cdots & p_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
p_{k1} & p_{k2} & \cdots & p_{kk}
\end{bmatrix}
\]

(2)

under the condition: \( 0 \leq p_{ij} \leq 1 \) and \( \sum_{j=1}^{k} p_{ij} = 1, (i, j = 1, 2, \cdots, k) \).

From this transition probability matrix, the probability from state \( i \) to state \( i+n \) after \( n \) steps can be observed directly.

In consideration of the regularity and simplicity of the tasks in a heterogeneous multi-core system, this paper designs a prediction model based on the Monte Carlo prediction theory, and the fundamental principles are shown in Figure 1 below.
Figure 1. Process of Memory Prediction and Allocation Algorithm.

The history information statistics $P$ is a localized two-dimensional array, which is used to record the transition information of every task's memory block. For example, the size of available memory blocks is $2^i (i \in [3,10])$, whose states set can be expressed as $\{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$, then

$$P_y = \text{Count}(S_i \rightarrow S_j)(i, j = 0, 1, \cdots, 7)$$  \hspace{1cm} (3)

$P_y$ is the transition frequency of a memory block from state $S_i$ to state $S_j$.

The prediction value is:

$$Pd_i = \left\{ S_j \mid S_{j-1} < \sum_{j=0}^{n} (P_y S_0 / \sum_{j=0}^{n} P_y) \leq S_j \right\}$$  \hspace{1cm} (4)

And the state change of the prediction chain is shown in Figure 2 below.

![Figure 2: Illustration of Memory Prediction Allocation Based on Statistics.](image-url)
UNION ALLOCATION MECHANISM

From the perspective of promoting prediction efficiency and reducing the segmentation frequency, this paper proposes a new union allocation mechanism which concentrates on reducing the fragment rate while ensuring the prediction accuracy and efficiency. Some fundamental concepts have to be introduced before the mechanism introduction.

The memory utilization is the actually used memory space divided by the original allocated space, the formula can be expressed as:

\[\eta = \frac{Used\ Space}{Allocated\ Space} \times 100\%\] (5)

Assuming \(\eta_1\) is the ratio of the memory utilization before applying this forecast-union allocation algorithm and \(\eta_2\) is the ratio after applying the algorithm, that is:

\[\theta = \frac{\eta_1}{\eta_2}\] (6)

From the two formulas above, because the actually used memory \(Used\ Space\) is changeless, thus \(\theta = \frac{Allocated\ Space_1}{Allocated\ Space_2}\), \(\theta\) is only related with the available memory block that can be allocated, and because the processes to be united is no more than 2, therefore \(\theta \in [1, 2]\).

The union allocation model is shown in Figure 3 below:

![Figure 3. Process of Memory Union Allocation.](image)

\(P_1 + P_2\) is the sum of actually used memory blocks of two processes, if adopting the original allocation mechanism shown by diagram(a), the memory blocks allocated to these two processes are \(Block_1\) and \(Block_2\), and therefore the memory utilization is \(\eta_1 = \frac{P_1 + P_2}{Block_1 + Block_2} \times 100\%\); if adopting the forecast-union allocation mechanism shown by diagram(b), the two processes can be regarded as one and be allocated a qualified block, and therefore the memory utilization is \(\eta_2 = \frac{P_1 + P_2}{Block_2} \times 100\%\). Since \(Block_1 > 0\), thus \(\eta_1 < \eta_2\), which means the forecast-union allocation mechanism is more efficient in promoting memory utilization and reducing fragment rate.
CONTRASTIVE EXPERIMENTS

Forecast Experiment

The test program is designed to run several prediction threads simultaneously, each one of them applies a memory block during its process of operation, and the accordingly prediction thread will conduct another memory prediction based on the current task's information.

The prediction thread:

```c
void predict_thread(INT8U prio, INT8U curX, INT32U x)
{
    ......
    for(;i<FL_SIZE;i++){
        //calculate the prediction value
        st = ptcb->stat[curX][i];
        esum += st * st;
        sum += st;
    }
    predsize = esum / sum + 1;
    total = power(2, no+1);
    if((predsize + x) < total){
        ptcb->blkinfo.BlkSize = total;
        ptcb->blkinfo.FirstSize = x;
        ptcb->blkinfo.SecondSize = predsize;
    } else {
        ptcb->blkinfo.BlkSize = 0;
        ptcb->blkinfo.FirstSize = 0;
        ptcb->blkinfo.SecondSize = 0;
    }
    PredTemp[prio] = predsize;
    PredictionList[predsize].task_size++;
    PredictionList[predsize].total++;
    return 0;
}
```

The measurement of a memory allocation algorithm is mainly about the hit ratio of memory block prediction, therefore a contrastive experiment is designed: dividing every 64 random numbers created by a random number generation function into 2 groups, on which applying two different prediction algorithms to make a comparison of two hit ratios, and the results of Matlab simulation experiments are shown in the table 1 below.

| Experiment Times 100 | 500 | 1000 | 2500 | 5000 |
|----------------------|-----|------|------|------|
| Original algorithm(%) | 35.5| 22.7 | 25.3 | 24.5 | 30.1 |
| Improved algorithm(%) | 22.4| 29.3 | 47.2 | 52.1 | 55.3 |

From the table above, along with the increase of cycle times, the hit ratio of the original algorithm changed little, while the hit ratio of the improved algorithm increased a lot, but the growth flattened as the cycle times increased to a certain extent.
Union Allocation Experiment

In order to analyze performance of the union allocation algorithm, μC/OS-II is selected as the experimental environment and transplanted to VC++. Since the memory management form of μC/OS-II is partition management, every memory block in the partition has the same size and cannot be divided any more, thus a memory information table BLOCK_INFO is added to recorded each fixed memory block's segmentation. Accordingly, “void *BlockInfo” variable is added into the original OS_MEM module:

typedef INT32U LEN;

typedef struct os_mem{
    LEN BlockSize;
    LEN Blocks;
    LEN BlockFree;
    LEN BlockName;
    void *Address;
    void *FreeList;
}OSBlkInfo;

The memory allocation function OSMemCreate() has to be modified accordingly:

OS_MEM *OSMemCreate(void *address,LEN blocks,LEN blocksize,LEN *err){
    ...
    OS_ENTER_CRITICAL();
    ...
    pMem->OSBlkInfo->BlkAddr=address;
    pMem->OSBlkInfo->BlockSize=totalSize;
    pMem->OSBlkInfo->FirstSize=currentSize;
    pMem->OSBlkInfo->SecondSize=predictionSize;
    OS_EXIT_CRITICAL();
    ...
}

Considering the complexity of memory allocation in a heterogeneous multi-core architecture, the dedicated test program CFRAC is selected to test the fragment rate based on the forecast-union allocation algorithm. As can be seen from the Figure 4 below, fragment rate of the forecast-union allocation algorithm is lower than the original algorithm, but it increases along with the allocated memory blocks.

Figure 4. Contrast of memory allocation.
CONCLUSION

Based on the research of the original memory allocation algorithm in heterogeneous multi-core architecture, the forecast-union memory allocation algorithm is designed, it consists of two parts: the first part is prediction of the required memory block based on history information statistics; the second part is union allocation of two separate memory blocks in which one block of them is predicted. The results of contrastive experiments show that the forecast-union memory allocation algorithm is relatively efficient in promoting timeliness and reducing fragment rate.

ACKNOWLEDGMENT

As the research of the thesis is sponsored by National Natural Science Foundation of China (No: 61063001, No: 61262075) and major scientific research project of Guangxi higher education (No: 201201ZD012), we would like to extend our sincere gratitude to them.

REFERENCES

1. Yan L., Shi Q., Chen T., et al. An On-chip Communication Mechanism Design in the Embedded Heterogeneous Multi-core Architecture [C]//Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on. IEEE, 2008: 1842-1845.
2. Lin-Dong L., De-Yu Q., Qiang C., et al. Efficient Scheduling Mechanism for Performance-Heterogeneous Multi-core Processor [C]//Digital Home (ICDH), 2014 5th International Conference on. IEEE, 2014: 342-346.
3. Wenjing L., Lisheng W. Energy-considered scheduling algorithm based on heterogeneous multi-core processor [C]//Mechatronic Science, Electric Engineering and Computer (MEC), 2011 International Conference on. IEEE, 2011: 1151-1154.
4. Okamoto T., Nakabayashi T., Sasaki T., et al. FabCache: Cache Design Automation for Heterogeneous Multi-core Processors [C]//Computing and Networking (CANDAR), 2013 First International Symposium on. IEEE, 2013: 602-606.
5. Yeh J.C., Ji K.M., Tung S.W., et al. Heterogeneous Multi-core SoC Implementation with System-Level Design Methodology [C]//High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on. IEEE, 2011: 851-856.
6. Marzi H., Yang Y. Microkernel based real-time embedded operating systems efficiency improvement [C]//Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on. IEEE, 2012: 1-4.