# 阿里内存故障隔离及专利

## 阿里的方案：内存故障的预测以及隔离

理论依据来源于 Google 的 paper 《DRAM Errors in the Wild: A Large-Scale Field Study》:

当在一个月内或者之前一个月内内存出现 CE 后，那么当月出现 UE 的概率将大幅提高。所以根据采集 CE 的信息，来预测 UE 的发生并提前进行隔离或者更换内存就有了实际意义。

好像好多互联网公司都在做这件事。

步骤一：CE 信息采集，可以直接通过 mcelog，或者金山目前的方案 skx_edac 驱动，也可以结合带外 log，或者 message，一起分析。如果是 mcelog，则需要格外的程序解析 mcelog 日志中的物理地址为具体的内存位置信息；skx_edac 驱动则直接给出了具体的内存位置信息。

步骤二：故障预测，mcelog 里面有简单的阈值漏斗算法来进行，而目前业界好像都引入了 AI 算法，通过相关算法，来预测相关内存是否会产生 UE；步骤三：根据预测结果，提前处理，mcelog 里面提供了多个脚本，最终会调用内核的接口，来实现隔离。

步骤四：目前 mcelog 里面没有的，内核的隔离操作并不都会成功，但是 mcelog 对于失败的情况并未进行额外的处理，可能阿里在这方面做了一些工作，在内核隔离失败后，返回给相应的应用程序错误信息，由应用程序触发虚机迁移操作，从而避免进一步的损失。

问题：

1. 并非所有的 UE 都可预测，有的 UE 触发时并不在之前必须产生 CE；所以覆盖率能有多大，需要系统部评估；
2. 通过 CE 预测 UE 只是在概率上有提升，并不能确定一定会发生，也就是结果有可能产生，也有可能没有产生； 所以准确率是一个问题；

| MCE类型 | | 动作 |
|---|---|---|
| dimm | uncorrected error | 当每个DIMM发生UC的次数超过阈值*uc-error-threshold*时，执行脚本*uc-error-trigger*。 |
| | corrected error | 当每个DIMM发生CE的次数超过阈值*ce-error-threshold*时，执行脚本*ce-error-trigger*。 |
| socket | mem uncorrected error | 当每个socket发生内存的UC错误超过阈值*mem-uc-error-threshold*时，执行脚本*mem-un-error-trigger*。 |
| | mem corrected error | 当每个socket发生内存的CE错误超过阈值*mem-ce-error-threshold*时，执行脚本*mem-ce-error-trigger*。 |
| | bus uncorrected error | 当总线UC错误发生时，执行脚本*bus-uc-threshold-trigger*。 |
| | IOMCA error | 当IOMCA UC发生时，执行脚本*iomca-threshold-trigger*。 |
| | unknown error | 当一些未知的错误发生时，执行脚本*unknown-threshold-trigger*。 |
| cache | cache error | Intel的CPU发发生cache错误超过阈值时，执行脚本*cache-threshold-trigger*。 |
| page | page corrected error | 1. 当同一个物理页面发生CE次数超过阈值*memory-ce-threshold*时，Off-lining (off, account, soft, hard, or soft-then-hard)该物理页。<br>2. 当同一个物理页年发生CE次数超过阈值*memory-ce-threshold*时，执行脚本*memory-ce-trigger*。 |

## 阿里的专利:

1. 其主要目的是，对于触发内核记录的的内存故障（可能是 UE 也可能是 CE），当然记录也可能是带外，根据发生故障的内存位置，按照一定的算法和逻辑推测其相邻的内存也可能会触发内存故障，从而提前隔离。对于内核来讲，目前已经提供了相关的接口给应用层（包括两个，soft_offline_page, hard_offline_page);
2. 此专利的逻辑是：对于已经发生故障的内存颗粒，其相邻的内存位置也很有可能出现错误 — 我不清楚这种逻辑是来源于阿里的统计还是内存厂商的建议，个人看来可能有些相关性，但是相关性有多大，没有具体的数据，不能确定，可能阿里仅仅就是申请了一个专利。
3. 如果上述逻辑属实，那么难点在于如何确定相邻的第二 page，当前通过内核可以拿到故障内存的具体位置，但是如何通过当前位置，寻找其周围的可能出问题的第二页是个问题，不太清楚内存里面具体的实现，可能与之相邻的 page 不止一个，如何确定目标的第二页，专利里面没有很详细的解释。
4. 如果已经确定目标第二页， 如何通过内存地址反推物理地址，我不太清楚如何实现，可能业界有已经写好的算法，这个 Intel 或者内存厂商可能有?
5. 由于缺乏具体的实验数据，我不太确定这个专利部署后会有效果。

## 相关材料

https://developer.aliyun.com/article/712012

http://www.cs.toronto.edu/~bianca/papers/sigmetrics09.pdf

2018109605671.pdf 1/1