Um estudo sobre o uso de memórias STT-RAM voláteis em diferentes níveis de memória cache

Giovane de O. Torres[1], **Rodrigo M. Duarte**[1]

[1]Centro de Desenvolvimento Tecnológico – UFPel
96.010-610 – Pelotas – RS – Brasil

{gdotorres,rmduarte}@inf.ufpel.edu.br

Resumo. *Atualmente existe a necessidade de melhorias na área de memórias, sendo o principal objetivo reduzir latência e consumo energético. Este trabalho faz uma avaliação de STT-RAM voláteis, uma tecnologia emergente de memória que pode suprir esta demanda. Porém, sua aplicação necessita de projetos elaborados para manter o correto armazenamento dos dados.*

1. Introdução

Com as diversas evoluções que ocorrem na área da computação, são sempre interessantes e importantes possíveis descobertas e inovações tecnológicas as quais podem alavancar melhorias em dispositivos eletrônicos. Estas melhorias podem incluir reduções tanto na latência como no consumo energético. Atualmente, uma das áreas que demandam estes aprimoramentos são as memórias – estas apresentam grande desafio na escalabilidade [Poremba and Xie 2012, Young et al. 2015] e desperdício de energia [Wang et al. 2013, Li et al. 2015]. Além disso, a memória é considerada como um dos componentes críticos em computadores [Perez and De Rose 2010, Zou et al. 2015]. Portanto, buscar alternativas que consigam extrair mais desempenho de memórias é essencial para o futuro.

Dentro deste contexto, tecnologias de memória emergentes visam melhorar o desempenho deste componente eletrônico. As MNVs (memórias não voláteis) estão inseridas dentro das novas tecnologias, sendo promissoras em diversos aspectos [Meena et al. 2014, Young et al. 2015, Zhao et al. 2015], os quais incluem:

- Prover baixo consumo energético;
- Melhor escalabilidade;
- Maior densidade por célula de memória;
- Garantia de armazenamento de informações na célula de memória sem a necessidade de efetuar operações de *refresh*.

Apesar dos diversos pontos positivos que as MNVs apresentam, as mesmas não são largamente utilizadas por apresentarem alguns desafios que devem ser superados. Um dos problemas que esta categoria de memórias apresenta está na durabilidade do material que compõem suas células, o qual é relativamente pequeno se comparados com tecnologias de memórias atuais, e.g. DRAM e SRAM [Mittal et al. 2015]. Outra dificuldade encontrada em MNVs ocorre quando deve-se fazer uma operação de escrita – tanto a latência como o consumo de energia nestas operações são considerados altos [Mittal and Vetter 2016].

Dentro do contexto de MNVs, existe diversos tipos de memórias as quais são diferenciadas pelos materiais que compõem suas células. Memórias estudadas dentro da bibliografia incluem: PCRAM (*Phase Change Random Access Memory*), STT-RAM (*Spin Transfer Torque Random Access Memory*), RRAM (*Resistive Random Access Memory*), MRAM (*Magnetic Random Access Memory*), FERAM (*Ferroelectric RAM*), além de outras memórias mais emergentes como *Racetrack Memory* e memória molecular [Meena et al. 2014]. Dentro deste amplo contexto, PCRAM, STT-RAM e RRAM são consideradas como NVMs que melhor podem compor sistemas de memória com grande eficiência e alta densidade, devido as propriedades melhoradas de escalabilidade e não volatilidade [Young et al. 2015], além de prometerem ser empregadas como memórias universais devido à capacidade de armazenamento e latências comparáveis a tecnologias como a DRAM [Mittal and Vetter 2016].

Para a melhor extração de desempenho de MNVs, existe diversas frentes de pesquisas que incluem modificações em arquiteturas bem como técnicas que possibilitem efetuar melhoras nas memórias. Uma das possibilidades incluem a redução da capacidade de retenção das MNVs com a finalidade de permitir que haja menor latência nas operações de memória, especialmente nas escritas. Esta técnica é chamada de **redução de retenção**, e pode ser aplicada ao utilizar-se das MNVs em *caches* ou na memória principal – isto porque estas memórias estão sujeitas a consideráveis quantidades de escritas, sendo os dados alterados com o passar do tempo. Com isto, reter informações por longos períodos de tempo sem modificações torna-se dispensável. A utilização de redução do tempo de retenção é vista como uma técnica potencial para difundir o uso de MNVs em sistemas de memória. Dentro do melhor do conhecimento dos autores, o redução de retenção teve até o presente momento seus efeitos testados na STT-RAM, sendo objeto de estudo de diversos trabalhos. No trabalho de [Mittal and Vetter 2016], a combinação de STT-RAM com redução de retenção é chamada de **STT-RAM volátil**.

Com isto, este trabalho tem como objetivo efetuar um estudo sobre as formas de utilização da STT-RAM volátil nos diferentes níveis dentro de uma hierarquia de memória – visando quais as melhores abordagens para o uso desta memória, bem como estudar a viabilidade da utilização de STT-RAM volátil. Verificou-se que as STT-RAM voláteis foram empregadas em diferentes níveis da hierarquia de memória, especificamente na parte de *caches*. Observou-se que o uso de STT-RAM volátil é promissor, desde que auxiliado por alguma técnica de *hardware* e/ou *software* que ajude na redução da quantidade de *refreshes*. Entretanto, estudos mais detalhados de sua aplicação são necessários para que isto torne-se viável.

2. STT-RAM e Volatilidade

A STT-RAM é uma memória conceitualmente não volátil e é considerada como uma das prováveis tecnologias que substituirão as SRAM [Kim and Choi 2016] – sendo estas últimas encontradas em memórias *cache*. As grandes vantagens das STT-RAM sobre as SRAM estão em: (i) na densidade, a qual pode ser quatro vezes maior [Zhou et al. 2009] e (ii) o menor consumo energético devido a não fuga de corrente [Wang et al. 2013]. Por outro lado, um dos principais problemas encontrados na STT-RAM é a estrondosa assimetria das operações de leitura e escrita em uma célula de memória – sendo que operações de escrita são bem mais custosas do que operações de leitura, tanto em energia quanto em tempo.

A STT-RAM é uma memória composta de dois elementos de armazenamento magnético, os quais são chamados de camada de referência e camada livre. Estes componentes magnéticos são posicionados um acima do outro juntamente com um terceiro elemento o qual é chamado de túnel de isolamento fino. A composição destes três elementos é conhecida como MTJ (*Magnectic Tunnel Junction*) [Zhou et al. 2009]. Já o efeito conhecido como *Spin Transfer Torque* é onde uma corrente pode efetuar uma movimentação brusca das camadas magnéticas que se encontram entre os materiais não-magnéticos. [Coalmon 2009]. Utilizando-se deste efeito e da diferentes orientações da magnetização torna-se possível o uso destes materiais como memória. O direcionamento da magnetização entre as camadas magnéticas resulta na diferença de resistência da MTJ. No momento que os campos magnéticos de ambas camadas estiverem paralelos, a resistência da MTJ é baixa, representando assim o valor 0 em uma célula de memória. Caso o campo magnético da camada livre possua uma direção oposta à da polaridade fixa, a resistência da MTJ torna-se alta, representando assim o valor 1 em uma célula de memória. A Figura 1 mostra esta visão conceitual da MTJ e como representar diferentes valores em uma célula STT-RAM.

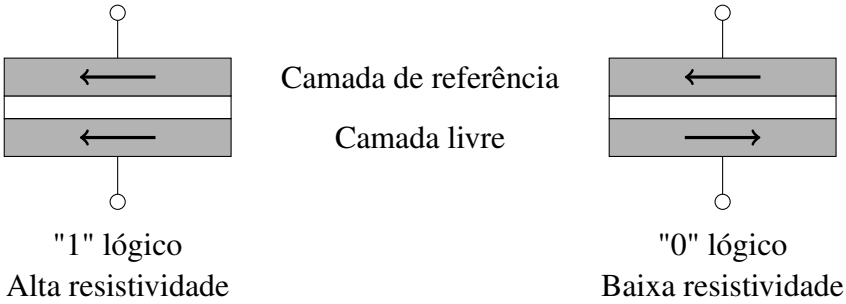

Figura 1. Visão conceitual da estrutura de MTJ [Zhou et al. 2009].

Com a finalidade de tornar o uso de memórias STT-RAM de maneira que operações de escrita em memória sejam menos impactantes, tanto no desempenho quanto no consumo energético, é a possibilidade de tornar a STT-RAM volátil. Isto é feito efetuando uma redução na área planar da MTJ da célula de memória – quanto menor a área, consegue-se escrever na célula STT-RAM de maneira mais rápida e que gaste menor quantidade de energia. Em contrapartida, a memória perde a sua capacidade de reter o dado por uma grande quantidade de tempo [Smullen et al. 2011], transformando-a assim em uma memória volátil, já que talvez necessitará de *refreshes* para não perder os seus dados, dependendo do quanto o tempo de retenção for reduzido. Com isto, têm-se células de memória com melhor escalabilidade e que conseguem amortecer o grande problema das MNVs em geral que é o tempo e o consumo energético de operações de escrita.

3. Trabalhos Relacionados

O primeiro trabalho que faz um estudo e apresenta o conceito de STT-RAM volátil é visto em [Smullen et al. 2011]. Este trabalho primeiramente explica como a STT-RAM pode ser volátil – efeito atingido através da redução da área planar da camada livre da MTJ da célula de memória. O artigo apresenta um modelo de STT-RAM volátil, exibe técnicas para otimização desta memória para prover melhorias no desempenho da escrita. Este modelo é testado em todos os níveis de *cache* nas questões de desempenho, energia

e *energy-delay*. Por fim, com a perda de retenção da STT-RAM, os dados em memória não podem ser perdidos. Com isto, o trabalho estuda um esquema simples de *refresh* que adiciona pouco *overhead* sobre o sistema de memória.

A primeira avaliação é efetuada sobre a relação entre as velocidades de leitura e escrita de uma célula STT-RAM. Chega-se à conclusão de que a fim de reduzir a latência das escritas, deve-se aumentar o tempo das leituras sem que isto deteriore o desempenho geral das leituras. Fazendo-se o processo reverso, i.e., aumentando o tempo de escrita numa célula da STT-RAM, a latência de leitura tende a diminuir. Outra análise é feita diminuindo a área planar da célula STT-RAM, sendo esta diretamente proporcional ao tempo de retenção do dado (Uma célula de $10F^2$ tem tempo de retenção de somente $56\mu s$). Ao reduzir o tempo de retenção, melhorias são percebidas na latência de escritas, bem como na energia em operações de leitura e escrita.

O trabalho faz então avaliações do impacto do uso da STT-RAM volátil através de simulações com *benchmarks*. Primeiramente, efetua-se a troca de todas as *caches* SRAM por STT-RAM voláteis, verificando que em questões de desempenho a maioria dos casos estudados existem perdas devido ao acréscimo nos tempos de escrita em memória. Em contrapartida, há uma redução de mais de 3x na fuga de energia ao usar STT-RAM. Outras avaliações que geraram resultados também foram realizadas, porém considerando hierarquias de memórias híbridas, podendo incluir tanto SRAM como os diferentes modelos de STT-RAM mostrados anteriormente. Os modelos de *cache* híbridos com STT-RAM de área $19F^2$ e $10F^2$ conseguem ter desempenho igual ou melhor comparado com *caches* SRAM. Nestes modelos híbridos, a redução do consumo energético é menor se comparada com *caches* puramente de tecnologia STT-RAM, porém ainda assim há ganhos consideráveis comparando com as SRAM.

Por fim, considerando que a memória estudada passa a ser volátil, o artigo propõe e utiliza-se de uma política de *refresh* simples, tal qual a DRAM, iterando por cada linha para aplicar um *refresh*. A última análise do trabalho envolve algumas configurações de hierarquias de *caches* as quais são testadas com este esquema de *refresh*, sendo comparadas com as hierarquias já visas anteriormente, bem como o *baseline* (*caches* SRAM). Conclui-se que ao utilizar *designs* de *caches* STT-RAM de $19F^2$ com *refresh* conseguem superar tanto no consumo quanto no desempenho os projetos de $32F^2$, além de prover melhor desempenho.

No trabalho apresentado em [Li et al. 2011], é realizado um *tradeoff* sobre o desempenho, confiabilidade e consumo energético das STT-RAM, aplicados aos requisitos no nível de arquitetura. No trabalho é apresentada a modelagem da STT-RAM para a redução do tempo de retenção dos dados. O objetivo é a aplicação em níveis mais específicos da hierarquia de memória, com foco em *caches on-chip*.

Primeiramente é feita uma análise para verificar quais são os impactos na quebra da não volatilidade da STT-RAM, para isso é realizada uma simulação de três diferentes desenhos do MTJ com 45x$95nm$ de tamanho. São realizadas duas otimizações da *baseline*, chamadas de Opt1 e Opt2. Sendo a *baseline* com seu tempo de retenção de dados de 4.27 anos e os outros dois otimizados para chaveamento (quando menor o tempo de retenção dos dados, menor o tempo necessário para chaveamento). Nos testes faz-se perceber que ao reduzir o tempo de retenção dos dados de 4 anos para $265\mu s$, a corrente decai de

	Cache de dados L1	*Cache* L2
Tamanho (Bytes)	32768	4194304
Associatividade	8	16
Tamanho do bloco (bytes)	64	64
Latência de Leitura (ciclos)	3	14

Tabela 1. Configuração das *caches* L1 de dados e L2, para a simulação.

$164.5\mu A$ para $71.4\mu A$ para um tempo de chaveamento de $10ns$ e temperatura de $350K$. Ao se colocar uma corrente de $125\mu A$ o tempo de chaveamento dos três MJTs varia de $29.8ns$ para $3.6ns$, um ganho de até 8 vezes.

É demostrando também a dependência da temperatura. A estabilidade da barreira magnética do MJT é sensível a temperatura de trabalho, assim, ao se aumentar a temperatura de $275K$ para $350K$ a corrente de chaveamento tende a diminuir. Logo após, o trabalho apresenta estatísticas de padrão de acesso a *cache*. Uma simulação é feita em um processador *quad-core* sobre as seguintes configurações de *cache* L1 de dados e L2:

O teste é realizado usando quatro aplicações do SPEC *benchmark* que são: 401.bzip2, 433.milc, 434zeusmp e 470lbm. Cada uma destas aplicações é executada em um *core* diferente, sendo simulado um total de 2 bilhões de instruções. Para a *cache* L1 mais de 95% dos dados são acessados nos primeiros 10^5 ciclos de *clock*, após isto, estes dados são carregados ou atualizados. Em alguns casos, como no *benchmark* 401.bzip2 o número pode alcançar até 99% e que comportamentos similares se apresentaram na *cache* L2. Estas observações demostram que uma pequena porção da *cache* de dados será ativa por longos períodos, como exemplo do *benchmark* 401.bzip2 que o tempo entre uma escrita e o último tempo em que os dados são lidos, excede 10^6 ciclos de *clock*. Este tempo entre leitura e escrita é utilizado para determinar o tempo mínimo de retenção de dados pelo MTJ.

No final é apresentado um modelo de *cache* híbrida conjunto associativa com n-vias, onde a mesma é dividida em dois modelos de MTJ, um com baixa retenção de dados (utilizando os dados apresentados anteriormente) e outra com alta retenção. Cada bloco da *cache* otimizada contém um contador que verifica quanto tempo os dados estão armazenados no bloco. Caso estes estejam por muito tempo, o sistema copia os dados da *cache* otimizada para a não otimizada. Para avaliar o desempenho, este novo modelo é comparado a um usando STT-RAM sem otimização. Os resultados mostram que usando metade das vias de uma *cache* L2 com 16 vias otimizada, o desempenho do novo modelo chega a ser 80% melhor. A conclusão é que a redução da não volatilidade da STT-RAM e o uso de uma *cache* híbrida, podem aumentar o desempenho e reduzir o consumo energético. Estas conclusões são retiradas da análise do modelo proposto e os testes realizados com os *benchmarks*.

Em [Sun et al. 2011] também é apresentada a proposta de implementação de *caches* híbridas, porém é apresentada a preocupação com a correta retenção dos dados na STT-RAM. Para se conseguir latências de escrita suficientemente baixas, é necessário reduzir muito o período de retenção dos dados, sendo que dependendo do tempo que o dado vai permanecer na *cache*, um processo de *refresh* dos dados acaba se tornando necessário. Assim é proposta a implementação de um sistema de *refresh* dinâmico. Para cada bloco da

cache é inserido um contador que monitora o tempo de retenção dos dados, se nenhuma escrita/leitura ocorre em um determinado período limite pré definido, um *refresh* ocorre. A ideia do uso do contador é para evitar a necessidade de *refreshes* desnecessários.

Uma *cache* com dois modelos de MTJ é proposto, um com baixa e outro com alta retenção. Os dados são movidos das regiões de alta para as de baixa retenção, ou vice versa, através das seguintes políticas:

- *Write Intensive*: se a intensidade de escrita é alta, e os dados se encontram em uma região de alta retenção, os dados são migrados para uma região de baixa.
- *Read intensive*: se os dados se encontram em uma região de baixa retenção, então estes são migrados para uma região de alta.
- *Neither write nor read intensive*: Os dados são mantidos em uma região de alta retenção ou migrados para memória principal.

A principal ideia para a migração de regiões de memória está ligada também a redução de *refreshes*, já que o modelo de *refresh* utilizado no modelo proposto é o mesmo utilizado em memórias DRAM.

Para os testes, são simulados três modelos de otimização de MTJ, um com baixa (lo), média (md) e alta retenção (hi). Em uma arquitetura *quad-core* com três níveis de *cache* L1, L2 e L3, ambas usando STT-RAM. Os resultados dos testes mostram que a melhor configuração para uma hierarquia de dois níveis (L1 e L2) é usando L1-lo e uma L2 hibrida com L2-lo para região de baixa retenção e L2-md para a região de alta. Já para uma *cache* de 3 níveis a melhor configuração se deu usando L1-lo, L2-lo com L2-md e una L3 com a configuração de uma L1-lo para a região de baixa retenção e L3-md para região de alta.

Os resultados dos testes mostram que o uso das *caches* híbridas propostas utilizando os métodos de *refresh* apresentados, tem um ganho de desempenho de até 99.8% se comparada ao uso de um SRAM convencional. Já no quesito consumo de energia, o modelo apresentado tem ganhos de até 70% se comparado a outros trabalhos e.g [Smullen et al. 2011]. O trabalho conclui que o uso de memórias híbridas (baixa e alta retenção) apresentam ganhos não somente em desempenho como também em consumo de energia. O uso de múltiplos níveis de retenção podem alcançar uma redução de até 73.8% de redução de energia se comparado a implementações que mesclam SRAM/STT-RAM e se comparado a outros trabalhos anteriores que apresentam o uso de STT-RAM, consegue um aumento de desempenho de até 5.5% e uma redução energética de até 30%.

O trabalho exibido em [Jog et al. 2012] tem o foco de ajustar o tempo de retenção de dado em memória de acordo com o tempo necessário para execução do *refresh* da *cache* de último nível (*Last Level Cache* – LLC), com a finalidade de obter ganhos no desempenho e energia. São propostos e estudados três modelos de STT-RAM: (i) um sem reduzir a retenção, (ii) outro o qual tem seu tempo de retenção reduzido para $1s$, considerado grande o bastante para o tempo de escrita da maioria das linhas de *cache*, não acarretando em *overhead*. E por fim, (iii) um modelo com tempo de retenção de $10ms$ o qual é necessário utilizar uma técnica de *refresh*, embora a latência e consumo de energia sejam menores para efetuar uma escrita neste caso.

A STT-RAM volátil neste caso funciona de maneira simplificada: Todos os blocos os quais após um determinado tempo estão por perder o dado, efetua-se um *write-back*

nos mesmos, utilizando-se de um contador de poucos *bits*. Esta abordagem acaba prejudicando o desempenho por que ao final do tempo de retenção, existirão diversos *write-backs*, causando um grande *overhead*. Além disto, se um bloco o qual é constantemente lido perder a informação, haverá maior quantidade de *misses* de leitura na *cache*. Para contornar principalmente o primeiro problema, é proposta a técnica de ***Cache Revive***, o qual adiciona um pequeno *buffer* que amortiza as custosas escritas que seriam feitas diretamente na STT-RAM.

Os resultados deste trabalho procuram avaliar, dentre os três modelos, algumas configurações de estudo de cache, as quais foram: S-1 (SRAM de $1MB$); S-4 (Caso hipotético onde a SRAM tem $4MB$, porém apresenta a mesma latência de operação de S-1); M-4 (STT-RAM não volátil); V-M-4($1s$) (STT-RAM volátil com tempo de retenção de $1s$); V-M-4($10ms$) (STT-RAM volátil com tempo de retenção de $10ms$) e R-M-4($10ms$) (STT-RAM volátil utilizando a técnica de *cache revive*).

Em questões de desempenho, em média o desempenho de R-M-4 ($10ms$) só perde para o modelo hipotético S-4, tendo *speedup* de quase 20% ao simular os *benchmarks* do PARSEC se comparados com o *baseline* S-1. Analisando o consumo energético, percebe-se um ganho de 44% na configuração M-4 para a S-1, o que é esperado dado que a STT-RAM gera pouca fuga de energia. Ao tornar a STT-RAM volátil, tem-se maior fuga de energia, ainda que inferior se comparada a SRAM. Na média, há ganhos em energia de 11% no caso de estudo R-M-4 ($10ms$) comparado com V-M-4($1s$), além de melhoria em 18% na energia em relação à configuração M-4 (*baseline* da STT-RAM), concluindo assim que energeticamente o uso de STT-RAM volátil com o esquema de *cache revive* atingiu os melhores resultados na média.

Em [Li et al. 2013], propõe-se uma técnica chamada de CCear (*Cache Coherence Enabled Adaptative Refresh*) a qual visa reduzir a quantidade de *refreshes* em uma STT-RAM volátil. Este trabalho tem como diferencial obsevar a informação de coerência dos blocos de *cache* L2 com a finalidade de atenuar o *overhead* gerado por operações de *refresh* na memória. Dentro da técnica de CCear, em cada bloco que é compartilhado são feitas n operações de *refreshes* após o carregamento da memória principal ou depois de um *write-back* da *cache* L1.

Utilizando-se do CCear, é feito um comparativo do uso de uma arquitetura com suporte a este mecanismo com a política de DRAM *refresh* proposta em [Smullen et al. 2011], sendo analisadas as questões energéticas e o IPC (Instruções por ciclo). Além destes dois casos, o trabalho ainda exibe resultados do que seria a situação ideal para a implementação de STT-RAM voláteis. Na energia, o modelo de DRAM *refresh* exibe consumo alto se comparado ao caso ideal, enquanto que ao usar o CCear há uma redução em relação à política de DRAM *refresh* na média em cerca de 10%. Avaliando o IPC, utilizando CCear existe uma melhora no IPC comparado ao DRAM *refresh* entre cerca de 3% a 7%, ocorrido devido à menor quantidade de conflitos entre *refreshes* e leituras na *cache*.

O artigo conclui dizendo que o *overhead* para armazenamento de informações necessárias para o funcionamento do CCear é desprezível - Inicialmente, é de menos de 0,2%, visto que cada bloco da LLC precisa de um bit para indicar se está expirado ou não. Além disto, cada bloco da LLC também usa um contador de 4 *bits* para controlar quantos

refreshes são feitos. Também conclui que o CCear pode adaptativamente minimizar a quantidade de operações de *refresh* necessárias para o uso de STT-RAM volátil.

Já o trabalho apresentado por [Chang et al. 2013], é realizada uma comparação entre três diferentes tipos de memórias aplicadas ao último nível de *cache*, são elas a SRAM, STT-RAM e eDRAM. A ideia é otimizar a SRAM para baixa corrente de fuga, a STT-RAM para baixo consumo energético de escrita e a eDRAM usando a ideia de *dead-line prediction* para reduzir a quantidade de *refreshes* desnecessários. A implementação proposta de STT-RAM neste trabalho é um MTJ com retenção de apenas 1 segundo, não são experimentados outros tempos de retenção de dados devido a necessidade de *buffers* e unidades de *refresh* para manter os dados corretamente armazenados. Os autores não se preocupam com isso porque o foco principal do artigo é o uso de eDRAMs.

Para os testes é simulada uma arquitetura com 8 *cores* e três níveis de *cache* onde, L1 e L2 são *caches* usando SDRAM de alto desempenho e L3 é uma *cache* de 32Mb, este último nível de *cache* com os três diferentes tipos de memória (SDRAM, STT-RAM e eDRAM). Os resultados mostram que o uso de STT-RAM consome até 48% menos energia que o uso de SRAM, contudo se a quantidade de escritas na memória é alta, STT-RAM consome maior quantidade de energia. Para cargas de trabalho com escrita intensiva a eDRAM se mostrou melhor na questão de consumo energético, ficando 36% mais eficiente que a SRAM de baixo consumo e 17% comparado a STT-RAM. No quesito desempenho, em média SDRAM apresenta os melhores tempos, porem em cargas de trabalhos dominadas por leituras, STT-RAM apresentam melhores resultados que as demais.

Outra observação feita no trabalho são os impactos que o tamanho e tecnologia exercem sobre as memórias. Caches que possuem alta densidade de memória usando STT-RAM tendem a apresentar menor consumo de energia. Isso se dá pelo fato de que quanto maior a *cache* em média, menor será o número de *cache miss*, sendo assim menor o número de atualizações. No quesito tecnologia, quanto menor a célula STT-RAM, menor sera o consumo de energia, pois o tempo de escrita será menor. STT-RAM com tecnologia de $22nm$ são mais eficientes energeticamente do que as de $45nm$, porém o tempo de retenção pode sofrer perda de estabilidade devido a instabilidade térmica.

Como conclusão os autores demostram que o uso de eDRAM com a técnica de *dead-line prediction* apresentou os melhores resultados, porém a necessidade de *caches* de desempenho elevado e alta densidade serão necessárias no futuro. Assim o uso de novas tecnologias como a eDRAM e a STT-RAM são promissoras.

No trabalho de [Li et al. 2015], também é proposta uma *cache* usando STT-RAM com baixo tempo de retenção de dados e com a necessidade de *refresh*. Porém o trabalho vai além, neste os autores se preocupam com a ordenação dos dados na *cache* para reduzir o número de *refreshes*. No trabalho é proposto uma ordenação dos dados em tempo de compilação para extrair os benefícios da STT-RAM e ao mesmo tempo reduzir o consumo de energia através da redução da quantidade de *refreshes* na memória. O trabalho propõe uma nova metodologia de *refresh* para memória chamado de N-*refresh*. Os dados que são escritos mas não são utilizados por um grande período de tempo ou sofrem uma reescrita, após um determinado tempo limite, são migrados para a memória principal, descartando assim a necessidade de *refresh* da *cache*. Também são propostos

dois modelos de organização de dado em memória no código compilado, são eles: ILP - *Integer Linear Programming*(ILP) e *Heuristic Data Layout*(DL), ambos utilizam técnicas sub-ótimas para ordenar os dados entre os blocos da *cache*.

Para os experimentos os autores simulam uma arquitetura de um processador *single-core* de 500Mhz e com uma *cache* de 16Mb, com blocos de 32b com 4 vias. Tempo de retenção de dados de $26.5\mu s$. O metodo de *refresh* e as técnicas de otimização em tempo de compilação são empregadas em outros dois metodos de *refresh* presentes na literatura [Sun et al. 2011, Jog et al. 2012].

Os resultados demostram que o uso da técnica N-*refresh*-DL, reduziu em apenas 2.0% a quantidade de *refreshes* se comparado a mesma técnica sem o uso de heurística (somente usando N-*refresh*). Porém os outros métodos já conseguiram melhores resultados se comparado a outros modelos da literatura, alcançando melhorias de até 38.0%.

Também é feito a análise da técnica apresentada a diferentes características da *cache* como tamanho, tamanho do bloco, latência de escrita, tempo de retenção e o mudanças no tempo de *refresh*. Para tamanhos de blocos pequenos, as técnicas apresentadas fornecem melhores desempenhos por explorar melhor a localidade dos dados, porém para *caches* com blocos pequenos a perda de desempenho devido a baixa taxa de *cache hit*, já para blocos grandes as técnicas não apresentam ganhos.

Em relação ao tamanho da *cache*, quanto maior a *cache*, maior se torna a quantidade de *refresh* necessários contudo, o método de organização dos dados, aproveita melhor o espaço da *cache*. Assim, aumentar o tamanho da *cache*, não surte efeito no desempenho, pois a nova área de memória quase não é utilizada. Já em relação ao aumento da quantidade de *refresh*, a taxa de *cache hit* aumenta pois menos blocos se tornam invalidados em decorrência da baixa retenção.

Os dois últimos itens (latência de escrita e retenção), quanto maior a latência de escrita maior é o tempo de execução do código e maior se torna a quantidade de *refresh* necessários, assim baixas latências de escrita apresentam melhores rendimentos. Quanto a retenção, quanto maior a retenção dos dados, menor são as taxas de *refresh* necessárias, porém a uma degradação no uso das técnicas de localidade de dados, pois a menos espaço para a otimização se torna limitado.

Como conclusão, o trabalho expõem que os métodos propostos apresentam melhor rendimento energético, por explorar melhor a localidade dos dados e diminuir a quantidade de *refresh* necessário e mantém um desempenho equivalente as técnicas apresentadas em outros trabalhos.

No trabalho de [Qiu et al. 2016], é proposto um método de escalonamento de *loops* para melhorar a localidade dos dados na memória e assim reduzir a quantidade de *refresh*. O trabalho propõem que os dados sejam acessados em uma determinada ordem, fazendo com que estes sejam "realimentados"por uma leitura e ou uma escrita. Isto se dá pelo fato de que, se o tempo de escrita/leitura é menor que o tempo de retenção dos dados na STT-RAM, o *refresh* se torna dispensável. Os autores defendem o fato de que alinhando os dados em memória em uma determinada forma, reduz-se a quantidade de energia consumida e aumenta-se o desempenho.

Os testes realizados foram a simulação de microcontrolador acessando uma

memória STT-RAM externa. Foram avaliadas três frequências de operação diferentes (100, 200 e 500MHz) e escalas de *loops* de 100x100(x100), 200x200(x200), 500x500(x500) e 1000x1000(x1000). Em média a técnica de escalonamento de *loops* os ciclos de acesso a memória (incluindo *refresh*) são reduzidos em 54.6%(100x100), 68.7%(200x200),82.7%(500x500) e 89.1%(1000x1000), em todas as frequencias testadas.

Já no consumo de energia dinâmica, as técnicas reduziram a energia consumida em 85.0% para 100Mhz, 75.8% em 200Mhz e 59.5% em 500Mhz. A explicação para esta redução de energia está no fato de que, quanto menor a frequência, maior se torna o tempo de escrita em memória, assim o espaço de otimização do *loop* é maior, alcançando melhores resultados. Outro item é o tamanho dos *loops*, quanto maior a escala do *loop* (1000x1000) maior se torna a distância entre uma leitura após uma escrita, assim a técnica de reescalonamento de *loops* apresenta melhores resultados, por reordenar estes acessos.

Uma outra observação é feita no trabalho. O quanto o tempo de retenção dos dados ajuda a reduzir o consumo de energia e aumenta o desempenho. Observa-se que quanto maior o tempo de retenção, o escalonamento de *loops* perde desempenho. Isso se dá pelo fato de que, o número de iterações do *loop* comparado ao tempo de retenção é muito menor, assim, mais atualizações se tornam necessárias na memória, prejudicando assim o uso da técnica.

Como conclusão, o trabalho expõe o problema do *overhead* criado pelos sistemas de *refresh* em memória, o que pode degradar tanto o desempenho quanto o consumo energético. Com o intuito de reduzir a necessidade de *refresh* em memória, o escalonamento de *loops*, para melhor a alocação e acesso de dados na memória se torna uma técnica promissora.

Por fim, o trabalho discutido em [Kim and Choi 2016] mostra uma abordagem sobre STT-RAM voláteis oposta ao dos trabalhos anteriores. Isto porque o artigo defende que a utilização de métodos de *refresh* em STT-RAM voláteis não são confiáveis o suficiente para serem utilizados. Afirma-se que o tempo de retenção da STT-RAM volátil não é determinístico, tampouco pode ser previsto. Além disto, falhas na retenção da STT-RAM volátil não podem ser previstos por causa da natureza estocástica da falha. Além disto, a STT-RAM volátil também apresenta três tipos de falhas, sendo estas:

- Escrita: Ao aplicar um pulso de corrente, este pode não ter tido tempo ou intensidade suficiente para alterar o dado;
- Leitura: Dependendo da intensidade da corrente aplicada a MTJ, esta pode realizar uma leitura errada ou até mesmo alterar o dado;
- Retenção: Não houve uma atualização dentro do tempo de vida da retenção do dado na MTJ.

Com isto, sugere-se a utilização de ECC (*Error Correction Code*) e *Scrubbing* para manter os dados em memória confiáveis. Por fim, uma célula STT-RAM volátil deve ser projetada de maneira cuidadosa no que diz respeito à sua estabilidade termal – para isso, deve-se levar em consideração: (i) o tamanho do *array* de memória, (ii) a força do código de ECC, (iii) o período de *scrubbing* e (iv) a confiabilidade-alvo.

A relação entre a estabiliadde termal e os quatro itens citados anteriormente é estudada neste trabalho. Alguns resultados são mostrados no sentido da relação entre

estes fatores, e.g. com o aumento do período de *scrubbing* de maneira exponencial, o mínimo de estabilidade termal aumenta lenta e linearmente. Além disto, ao usar ECC, reduz-se o mínimo de estabilidade termal comparada a STT-RAM não volátil. Outras observações vistas foram:

- Quanto mais frequente o *scrubbing*, mais ineficiente ele tende a ser, dado que o mesmo diminui pouco a estabilidade termal necessária enquanto que o *overhead* gerado pelo *scrubbing* aumenta significativamente em termos de desempnho e energia;
- Técnicas de ECC mais simples são melhores para redução da estabilidade termal;
- A medida que diminui a taxa de falha exponencialmente, a estabilidade termal também aumenta lentamente de maneira linear.

Outros resultados são exibidos verificando o desempenho e a eficiência energética ao usar STT-RAM volátil com *scrubbing* e ECC. A STT-RAM é empregada como LLC, e são testadas 9 diferentes configurações envolvendo o tempo de retenção da memória e diferentes níveis de ECC. Chega-se a conclusão de que a utilização de tanto ECC como *scrubbing* faz com que a STT-RAM volátil torne-se ineficiente em termos de desempenho e energia de maneira geral. Somente o uso da técnica mais simples de ECC com um período de *scrubbing* maior que $10ms$ consegue ter redução de consumo energético comparada à *baseline* (STT-RAM não volátil).

4. Discussões

As STT-RAM voláteis, dentro dos artigos estudados, mostraram-se aplicáveis como futuras substitutas para as SRAM. Isto se deve a diversos fatores os quais foram avaliados nos trabalhos, como o estudo da utilização em vários níveis de *cache*, consumo energético reduzido (tanto comparavéis com a STT-RAM não volátil, quanto com as SRAM) e maior densidade para uma mesma área de memória se comparada a SRAM.

O principal impedimento para uso de STT-RAM a nível de memórias *cache* é a alta latência de escrita. Com o intuito de mitigar este problema, utilizam-se técnicas como a redução da área da MTJ, aumento da temperatura de trabalho e principalmente a redução da corrente necessária para efetuar uma escrita em memória. Observando os projetos de células STT-RAM avaliados nos trabalhos estudados, foi possível verificar que existem *tradeoffs* entre diversas características físicas. A Figura 2 mostra a relação entre tempo de retenção, latência de escrita, estabilidade, corrente de escrita e taxa de *refresh*.

Tomando como base projetos completamente opostos de STT-RAM, o gráfico demonstra que ao reduzir a área da MTJ, consegue-se uma redução na latência e corrente de escrita. Em contrapartida, há a redução da retenção dos dados e da estabilidade da MTJ, tornando-se assim necessária a utilização de *refresh*. Já com uma MTJ de maior área, obtemos maior retenção dos dados e melhor estabilidade, sem a necessidade de *refresh*. Contudo, aumenta-se a latência e corrente de escrita. A decisão de utilizar um determinado tamanho de área para MTJ vai da necessidade de projeto empregado.

Dentro do contexto das STT-RAM que necessitam de *refresh* para manter a corretude dos dados, os artigos estudados utilizaram-se de técnicas auxiliaries tanto em *hardware* como em *software*. Os mecanismos avaliados em *hardware* incluíram desde *refreshes* estilo DRAM, hierarquias de *caches* híbridas (SRAM, STT-RAM tanto volátil

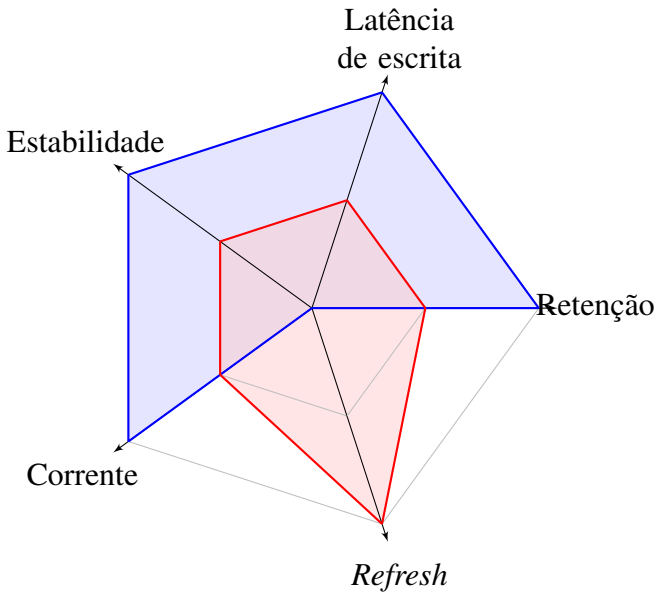

Figura 2. *Tradeoffs* entre as caracaterísticas da célula STT-RAM.

quanto não volátil) e o uso de contadores que definem a necessidade ou não de fazer *refresh*. Já em *software*, uma das técnicas ordena os dados em tempo de compilação para evitar *refreshes* desnecessários. Estas utilizam heurísticas para prever dados que serão acessados graças à localidade temporal. Outra possibilidade encontrada foi o escalonamento dos dados com a finalidade de explorar a localidade especial, o que também reduz a quantidade de *refreshes*.

Apesar das técnicas apresentadas acima proverem meios para o uso de STT-RAM voláteis, e manter a premissa de baixo consumo energético, um estudo apresentado em [Kim and Choi 2016] exibe problemas na utilização de *refreshes* estilo DRAM para manter os dados nestas memórias atualizados. Isto porque uma falha na retenção de STT-RAM volátil não pode ser evitada usando *refresh* estilo DRAM devido à natureza estocástica desta falha. Além disto, existem três tipos de falha que a STT-RAM volátil pode apresentar, sendo que estas podem ocorrer nas operações de leitura, de escrita e na retenção do dado. Sugeriu-se o uso de ECC e *scrubbing* para o uso de STT-RAM volátil, o qual devido ao *overhead* gerado por estas técnicas, não obteve bons resultados tanto em desempenho quanto em consumo energético.

5. Conclusões

Este trabalho fez um estudo acerca do uso de STT-RAM voláteis em diferentes níveis de *cache*. A ideia principal foi fazer uma análise de quais as técnicas empregadas para a otimização o uso destas memórias. Para isto, diversos trabalhos relacionados foram estudados, os quais demonstraram diversos métodos que variam do *hardware* até o *software*. Os resultados apresentados demonstram que a utilização de STT-RAM volátil é promissora como substituta das SRAM, devido a dois principais fatores: Baixo consumo energético e alta densidade de memória. Contudo, estudos mais recentes apontaram que uma investigação mais aprofundada a respeito de STT-RAM voláteis deve ser feita – isto porque diversas falhas foram observadas nas operações de memória, o que incluem es-

critas e leituras, além de que falhas na retenção do dado da STT-RAM possuem natureza estocástica. Com isto, técnicas mais aprimoradas de *refresh* tornam-se necessárias para que a utilização de STT-RAM voláteis tornem se viáveis.

Referências

Chang, M. T., Rosenfeld, P., Lu, S. L., and Jacob, B. (2013). Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 143–154.

Coalmon, B. (2009). Theory of spin transfer torque. 2009. Disponível em: <https://www.nist.gov/programs-projects/theory-spin-transfer-torque>. Acesso em: novembro de 2016.

Jog, A., Mishra, A. K., Xu, C., Xie, Y., Narayanan, V., Iyer, R., and Das, C. R. (2012). Cache revive: Architecting volatile stt-ram caches for enhanced performance in cmps. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 243–252, New York, NY, USA. ACM.

Kim, N. and Choi, K. (2015). A design guideline for volatile stt-ram with ecc and scrubbing. In *2015 International SoC Design Conference (ISOCC)*, pages 29–30.

Kim, N. and Choi, K. (2016). Exploration of trade-offs in the design of volatile stt–ram cache. *Journal of Systems Architecture*, 71:23 – 31.

Li, H., Wang, X., Ong, Z. L., Wong, W. F., Zhang, Y., Wang, P., and Chen, Y. (2011). Performance, power, and reliability tradeoffs of stt-ram cell subject to architecture-level requirement. *IEEE Transactions on Magnetics*, 47(10):2356–2359.

Li, J., Shi, L., Li, Q., Xue, C. J., Chen, Y., and Xu, Y. (2013). Cache coherence enabled adaptive refresh for volatile stt-ram. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1247–1250, San Jose, CA, USA. EDA Consortium.

Li, Q., He, Y., Li, J., Shi, L., Chen, Y., and Xue, C. J. (2015). Compiler-assisted refresh minimization for volatile stt-ram cache. *IEEE Transactions on Computers*, 64(8):2169–2181.

Meena, J. S., Sze, S. M., Chand, U., and Tseng, T.-Y. (2014). Overview of emerging nonvolatile memory technologies. *Nanoscale research letters*, 9(1):1.

Mittal, S. and Vetter, J. S. (2016). A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1537–1550.

Mittal, S., Vetter, J. S., and Li, D. (2015). A survey of architectural approaches for managing embedded dram and non-volatile on-chip caches. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1524–1537.

Perez, T. and De Rose, C. (2010). Non-volatile memory: Emerging technologies and their impacts on memory systems (tr-060). Technical report, Technical report, Faculdade de Informática, Pontificia Universidade Católica do Rio Grande do Sul (PUCRS), Porto Alegre, Brazil.

Poremba, M. and Xie, Y. (2012). Nvmain: An architectural-level main memory simulator for emerging non-volatile memories. In *2012 IEEE Computer Society Annual Symposium on VLSI*, pages 392–397. IEEE, IEEE.

Qiu, K., Luo, J., Gong, Z., Zhang, W., Wang, J., Xu, Y., Li, T., and Xue, C. J. (2016). Refresh-aware loop scheduling for high performance low power volatile stt-ram. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pages 209–216.

Smullen, C. W., Mohan, V., Nigam, A., Gurumurthi, S., and Stan, M. R. (2011). Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 50–61.

Sun, Z., Bi, X., Li, H. H., Wong, W.-F., Ong, Z.-L., Zhu, X., and Wu, W. (2011). Multi retention level stt-ram cache designs with a dynamic refresh scheme. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 329–338, New York, NY, USA. ACM.

Wang, K., Alzate, J., and Amiri, P. K. (2013). Low-power non-volatile spintronic memory: Stt-ram and beyond. *Journal of Physics D: Applied Physics*, 46(7):074003.

Young, V., Nair, P. J., and Qureshi, M. K. (2015). Deuce: Write-efficient encryption for non-volatile memories. *SIGARCH Comput. Archit. News*, 43(1):33–44.

Zhao, J., Xu, C., Chi, P., and Xie, Y. (2015). Memory and storage system design with non-volatile memory technologies. *IPSJ Transactions on System LSI Design Methodology*, 8(0):2–11.

Zhou, P., Zhao, B., Yang, J., and Zhang, Y. (2009). Energy reduction for stt-ram using early write termination. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 264–268. ACM, ACM.

Zou, Q., Poremba, M., He, R., Yang, W., Zhao, J., and Xie, Y. (2015). Heterogeneous architecture design with emerging 3d and non-volatile memory technologies. In *The 20th Asia and South Pacific Design Automation Conference*, pages 785–790. IEEE.