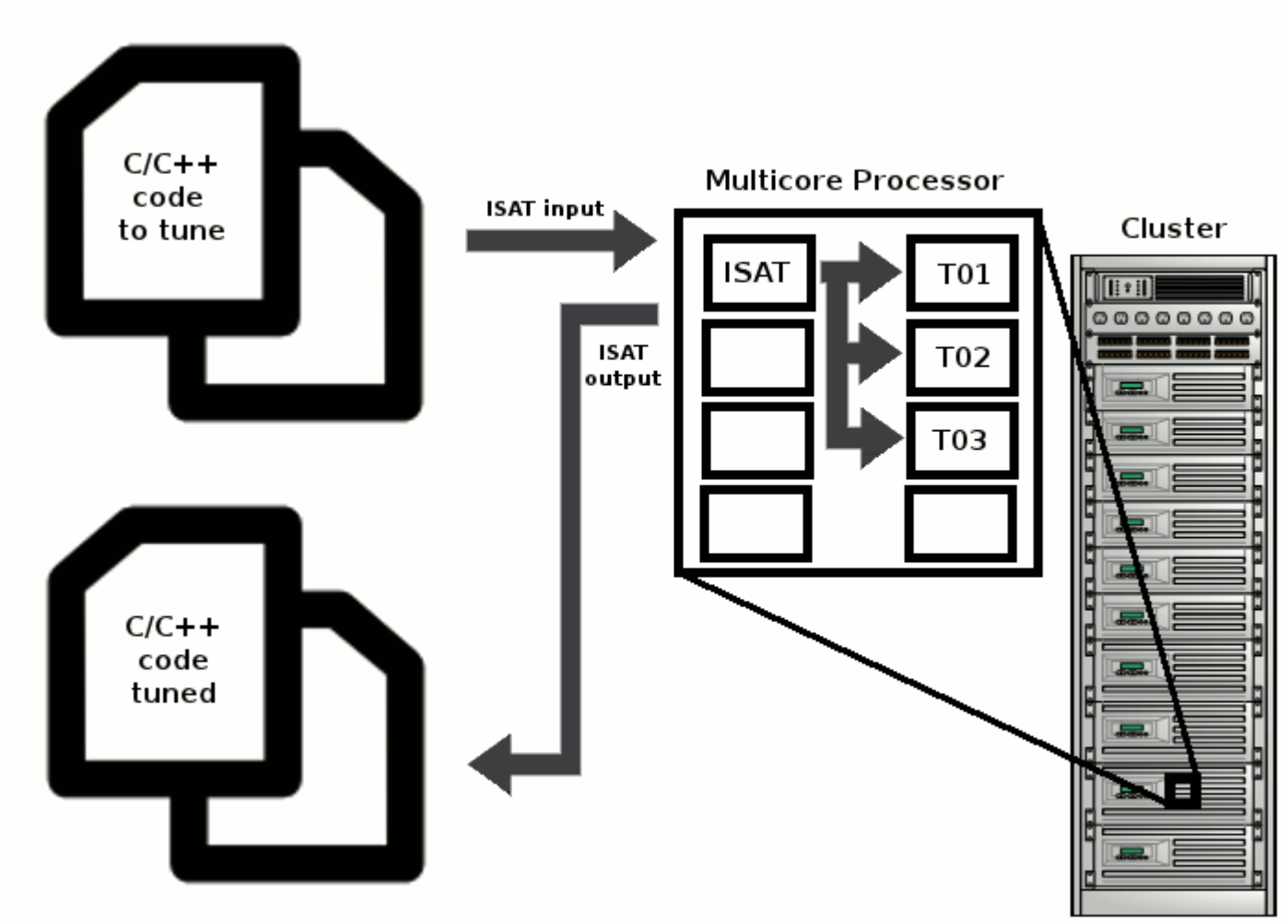
Aplicação de Técnicas de *Auto-Tuning* em Memórias Transacionais

Paulo Vital e Rodolfo Azevedo

Instituto de Computação

UNICAMP

Motivação


C/C++
code
to tune
ISAT input
Multicore Processor
Cluster
ISAT
T01
T02
T03
ISAT
output
C/C++
code
tuned

Agenda

- Motivação
- Memórias Transacionais e *Auto-Tuning*
- ISAT
- Proposta de trabalho
- Conclusões e Perguntas

Memórias Transacionais

- Paradigma de programação paralela
- Foco no problema de sincronização de dados
- Conceito de transação:
  - Atomicidade
  - Consistencia
  - Isolamento
- [Larus and Rajwar 2006] definem três tipos:
  - Software Transactional Memory - STM
  - Hardware Transactional Memory - HTM
  - Hybrid Transactional Memory - HyTM

Auto-Tuning

- Encontra melhor implementação ou valor ótimo para um ou mais parâmetros e/ou algoritmo
- [Williams 2008], apresentam três conceitos:
  - Espaço de otimização
  - Geração de código
  - Exploração
- Boa técnica para produzir programas eficientes e portáveis:
  - Configuração ajustada
  - Cluster ou nuvem

ISAT

- *Intel® Software Autotuning Tool* (ISAT)
- busca automática de valores próximos do ideal
- saída é uma versão ajustada do programa
- provê visualização gráfica dos resultados
- Utiliza pragmas no formato “**#pragma isat ....** ”
  - marker
  - tuning
- Busca independente ou dependente

ISAT

```
static void BlockedMatrixMultiply(FLOAT* A, FLOAT* B, FLOAT* C, const int n)
{
#pragma isat tuning name(tune_mm) scope(M1_begin, M1_end) measure(M2_begin, M2_end)
variable(blk_k, range($L1_CACHE_LINESIZE, 16*$L1_CACHE_LINESIZE, $L1_CACHE_LINESIZE/2))
variable(blk_j, range($L1_CACHE_LINESIZE, 16*$L1_CACHE_LINESIZE, $L1_CACHE_LINESIZE/2))
variable(blk_i, range($L1_CACHE_LINESIZE, 16*$L1_CACHE_LINESIZE, $L1_CACHE_LINESIZE/2)) //
add "search(dependent)" at the end of this pragma if you want to search the variables
dependently

#pragma isat marker M1_begin

    const int blk_i= 64;
    const int blk_j= 64;
    const int blk_k= 64;

    for (int i=0; i<n; i += blk_i)
        for (int j=0; j<n; j += blk_j)
            for (int k=0; k<n; k += blk_k)
                for (int ii=i; ii<MIN(i+blk_i, n); ii++)
                    for (int jj=j; jj<MIN(j+blk_j, n); jj++)
                        for (int kk=k; kk<MIN(k+blk_k, n); kk++)
                            C[ii*n+jj] = C[ii*n + jj] + A[ii*n+kk]*B[kk*n+jj];
#pragma isat marker M1_end
}
```

ISAT

```
main(int argc, char** argv)
{
    FLOAT* A = new FLOAT[N * N];
    assert(A);

    FLOAT* B = new FLOAT[N * N];
    assert(B);

    FLOAT* C_blocked = new FLOAT[N * N];
    assert(C_blocked);

    Initialize(A, N, 1);
    Initialize(B, N, 2);
    Initialize(C_blocked, N, 0);

#pragma isat marker M2_begin
    BlockedMatrixMultiply(A, B, C_blocked, N);
#pragma isat marker M2_end

    const FLOAT s_blocked = Sum(C_blocked, N);

    cout << "Sum-blocked = " << s_blocked << endl;
}
```

Proposta de Trabalho

- Trabalho em desenvolvimento
- Proposta é criar uma ferramenta de auto-tuning com suporte a STM
- Baseado no ISAT:
  - Suporte a busca paralela;
  - Nova condição de seleção;
  - Suporte a STM.

Proposta de Trabalho

- ISAT é muito linear;
- Problema: explosão combinatória;
- Suporte a busca paralela:
  - Redução do tempo de busca:
    - Busca dependente: várias threads para cada parâmetro
    - Busca independente: várias threads independente dos parâmetros
  - Sistemas multiprocessados (multicore)
  - Cluster/Nuvem

Proposta de Trabalho

```
#pragma isat tuning name(tune_test) scope(M1_begin, M1_end) measure(M2_begin, M2_end)
variable(x, [10, 20, 30]) variable(y, [70, 80, 90, 100])) search(independent)
```

- Busca independente: 3+4
- Busca dependente:
  - (10, 70), (10, 80), (10, 90), (10, 100),
  - (20, 70), (20, 80), (20, 90), (20, 100),
  - (30, 70), (30, 80), (30, 90), (30, 100)

Proposta de Trabalho

- Comparação de valores em loops de execução

```
def FindOptTvalFromFeedbacks(self):
    tvalList = self._feedbackInfoDict.TvalList()
    numTvals = len(tvalList)
    AssertAlways(numTvals > 0, 'No feedback collected for this tvar\n')

    first = True
    opt_tval = None
    opt_time = None
    for i in range(0, numTvals):
        tval = tvalList[i]
        fb = self._feedbackInfoDict.GetInfo(tval)
        time = fb.MedianTime()
        if first:
            opt_tval = tval
            opt_time = time
            first = False
        else:
            if time < opt_time:
                opt_tval = tval
                opt_time = time

    AssertAlways(opt_tval, 'No opt_tval found\n')
    AssertAlways(opt_time, 'No opt_time found\n')

    self._optTval = opt_tval
```

Proposta de Trabalho

- Adição de novo processo de seleção baseado em algoritmo genético:
  - Indivíduo ótimo de uma população geneticamente refinada através de gerações
  - População inicial baseado no intervalo definido no #pragma tuning
  - Objetivo: código mais eficaz e reduzir provável explosão combinatória

Proposta de Trabalho

- ISAT suporta:
  - OpenMP
  - Intel TBB
- Adição de suporte a STM:
  - TL2 (x86 e x86_64)
  - *Transactional Software Cache* (TSC)
- Testes:
  - Cluster/Nuvem
  - Sistemas Assimétricos: Cell BE
  - STAMP e modificações

Conclusões

- Trabalho em desenvolvimento
- Soma de tópicos atuais:
  - Desempenho em STM
  - Utilização de *auto-tuning*
- Objetivos:
  - Reforçar o uso de *auto-tuning*
  - Demonstrar com a “soma”:
    - Aumento de desempenho
    - Transparência para o desenvolvedor
    - Eficiência e rapidez na configuração

Perguntas ???