Universidade Federal de Santa Catarina

DAS5306: Programação Concorrente e Sistemas de Tempo Real

Relatório T2

Trabalho de Tempo Real

Alunos
Iago de Oliveira Silvestre
Ígor Assis Rocha Yamamoto

Professor
Rômulo Silva de Oliveira

Junho de 2016

Sumário

1 Código

Para a medição do tempo de resposta dos controladores de nível e temperatura, foram adicionadas funções *clock_gettime()*, antes e após a chamada das funções dos controladores (*controla_nivel()* e *controla_temperatura()*). A variável *t*, guarda o tempo de início do ciclo de controle, enquanto as variáveis *tn* e *tt* recebem o tempo de término das atividades de controle de nível e temperatura, respectivamente.

Após a obtenção dos tempos de início e término, o tempo de resposta é calculado através da função diff(struct timespec inicio, struct timespec fim) e adicionado ao buffer de tempos para o nível e temperatura. A cada coleta de 100 dados de amostra (NSAMPLE=100), é feita uma escrita em arquivo com os dados dos buffers, que é esvaziado. No total são feitas 100 amostras de 100 dados, totalizando 10000 tempos de resposta para cada controlador.

A seguir é apresentado parte do código modificado para realizar a medição dos tempos de resposta.

```
int main()
{
long aux=0;
int it=0, in=0; count=0;
long interval = 10000000; /* 10ms*/ // sensores
long t_resposta;
    struct timespec t,tt,tn;

... codigo de inicializacao ...

    while(count<100) {
        clock_nanosleep(CLOCK_MONOTONIC, TIMER_ABSTIME, &t, NULL);
        clock_gettime(CLOCK_MONOTONIC, &t);
        if(aux%7==0 & in<NSAMPLE){
                atuador("anf",0);
                controle_nivel(); //malha controle nivel
                clock_gettime(CLOCK_MONOTONIC, &tn);
                t_resposta = dif(t, tn);
                tempo_N[in] = t_resposta;
                in++;

        }
        if(aux%9==0 & it<NSAMPLE){
                controle_temperatura(); //malha controle temperatura
                clock_gettime(CLOCK_MONOTONIC, &tt);
                t_resposta = dif(t, tt);
                tempo_T[it] = t_resposta;
```

```
                it++;
        }
        if(aux%100==0){
                atualiza_tela();
        }
    if(it==NSAMPLE & in==NSAMPLE){
        it=0;
        in=0;
        armazena_tempos();
        count++
        }
            t.tv_nsec += interval;
        while (t.tv_nsec >= NSEC_PER_SEC) {
                t.tv_nsec -= NSEC_PER_SEC;
                t.tv_sec++;
            }
        aux++;

  }
exit(EXIT_SUCCESS);
}
```

```
long dif(struct timespec inicio, struct timespec fim){
        long temp;
        temp = (fim.tv_sec*NSEC_PER_SEC+fim.tv_nsec) -\
        (inicio.tv_sec*NSEC_PER_SEC+inicio.tv_nsec);
        //temp = temp/NSEC_PER_MSEC;
        return temp;
}
```

```
void armazena_tempos(){
int i;
FILE *pf;
if((pf = fopen("TemposNivel.txt","a")) == NULL){
/* Abre arquivo binario para escrita */
        printf("Erro na abertura do arquivo");
        exit(1);
}
//fprintf(pf, "Tempos de Resposta - Controlador Nivel \n");
for(i=0;i<NSAMPLE;i++) {
        fprintf(pf, "%ld\n",tempo_N[i]);
        tempo_N[i] = '\0';
}
fclose(pf);
if((pf = fopen("TemposTemperatura.txt","a")) == NULL){
/* Abre arquivo binario para escrita */
        printf("Erro na abertura do arquivo");
        exit(1);
}
```

```
//fprintf(pf, "\nTempos de Resposta - Controlador Temperatura \n");
for(i=0;i<NSAMPLE;i++) {
        fprintf(pf, "%ld\n",tempo_T[i]);
        tempo_T[i] = '\0';
}
fclose(pf);
}
```

2 Coleta e Análise de Dados

A partir do código descrito anteriormente, foram realizadas 2 coletas de dados no sistema operacional linux: uma feita com baixa atividade de processos sendo executados na máquina; outra feita sob stress do sistema.

A coleta feita em baixa atividade do sistema foi feita simplesmente executando o código do programa em c e o simulador em java da caldeira. Enquanto para realizar a coleta em alta atividade do sistema, foi utilizado o comando "stress –cpu 8 –io 4 –vm 2 –hdd 1"do linux (criação de 8 processos cpu-bound, 4 processos io-bound, 2 processos de alocação de memória e um processo de uso de disco).

Após a coleta dos dados, foi feita uma análise gráfica dos tempos de resposta para cada controlador. Com o uso do pacote matemático Octave, foram criados histogramas através da função *hist()* e também foram feitos gráficos da função de densidade de probabilidade a partir dos valores médios e do desvio padrão de cada amostra para os dois controladores.

2.1 Controle de Nível

2.1.1 Histogramas

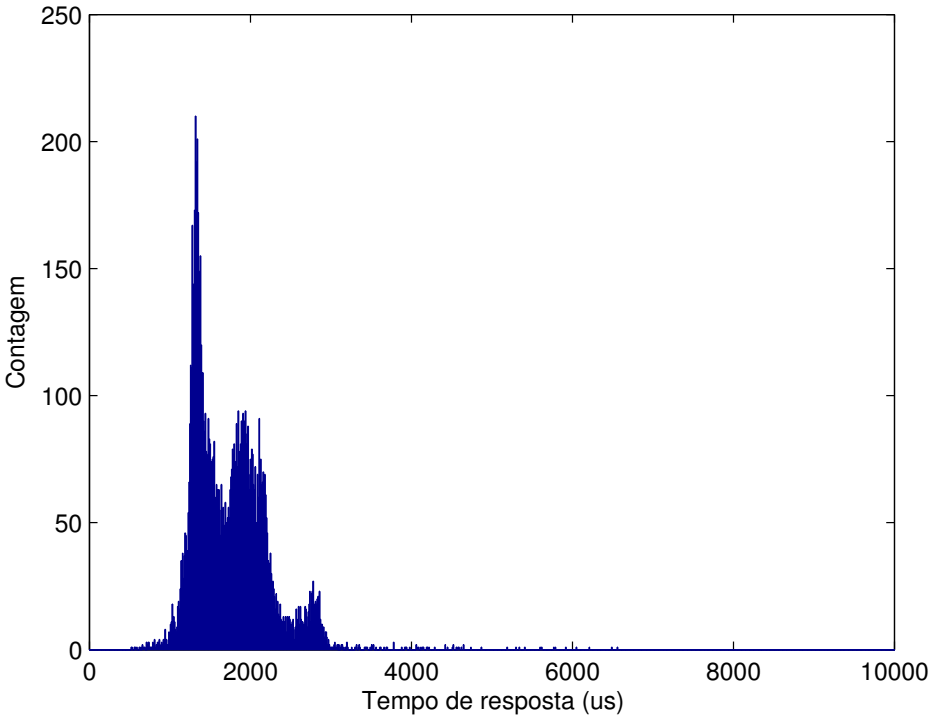

Figura 1: Histograma de Respostas Nível em Baixa Atividade

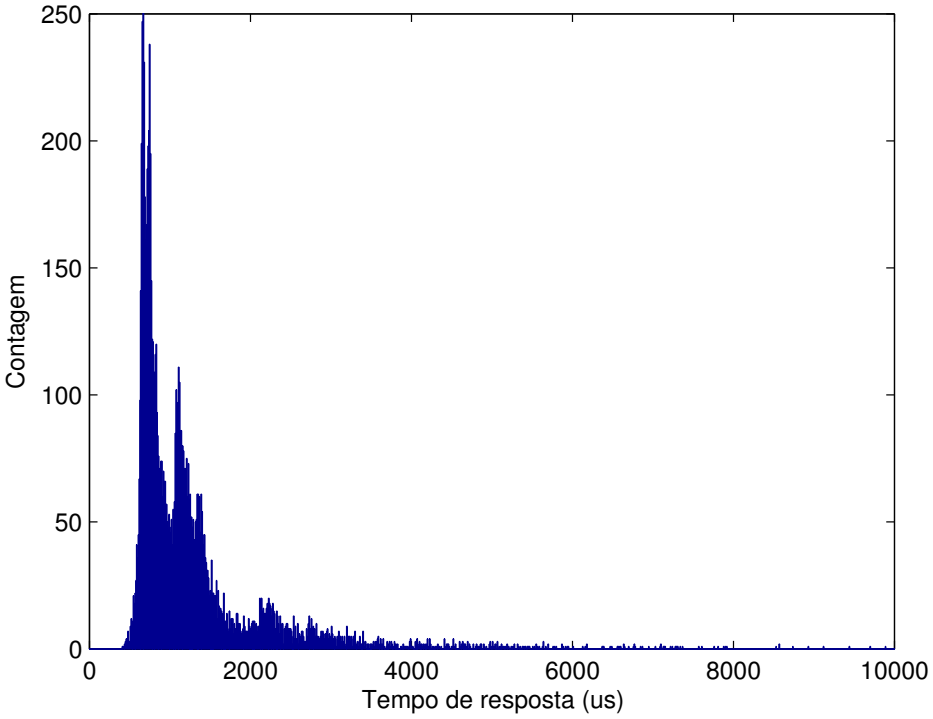

Figura 2: Histograma de Respostas Nível em Alta Atividade

2.1.2 Distribuição Normal de Probabilidade

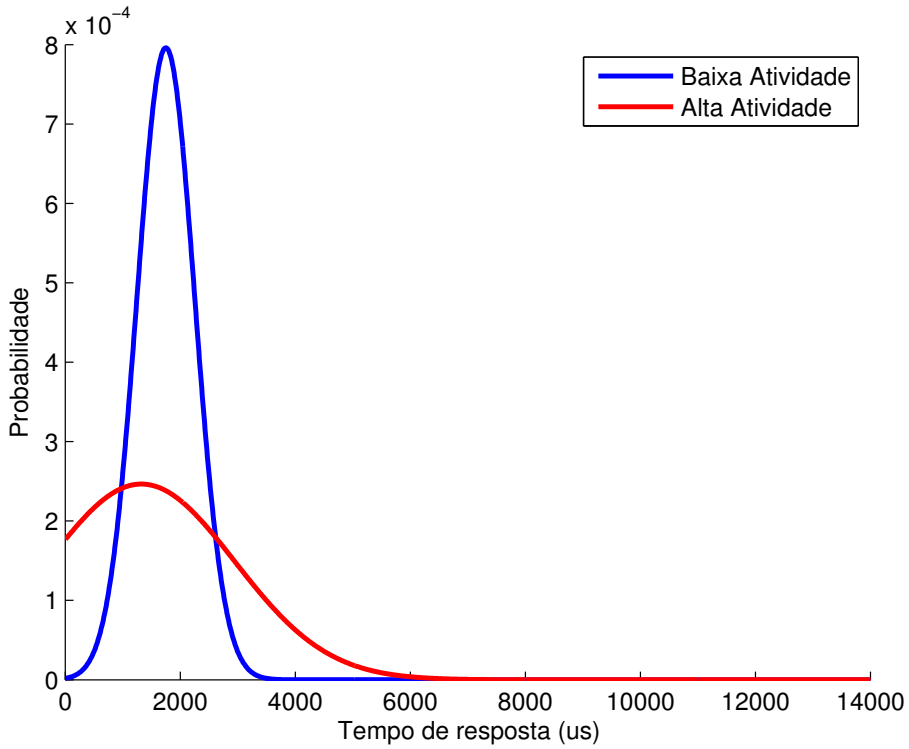

Figura 3: Função Densidade de Probabilidade Nível

2.2 Controle de Temperatura

2.2.1 Histogramas

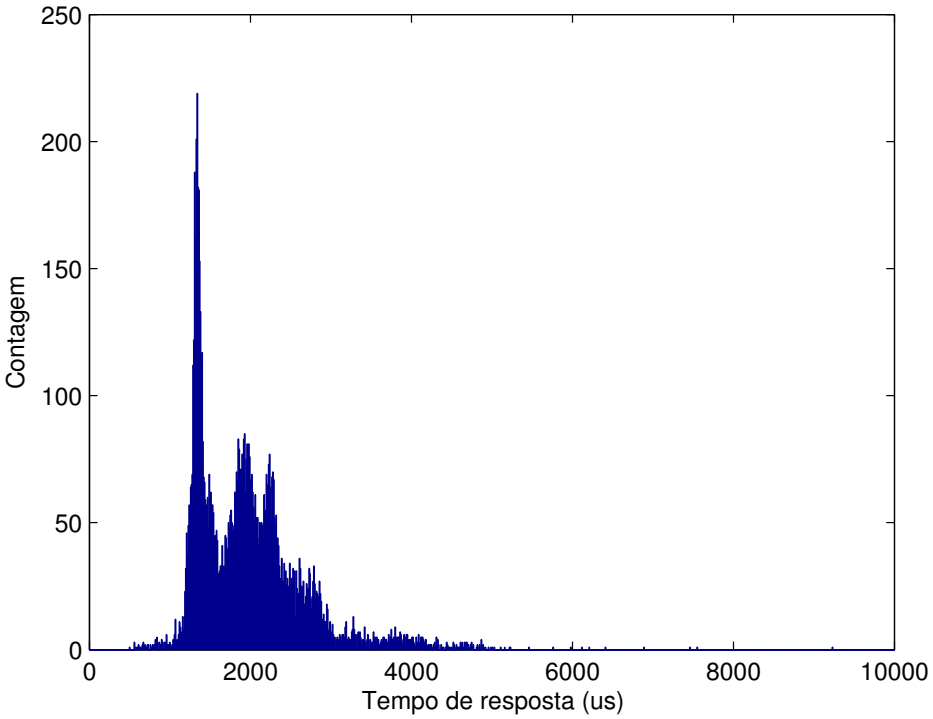

Figura 4: Histograma de Respostas Temperatura em Baixa Atividade

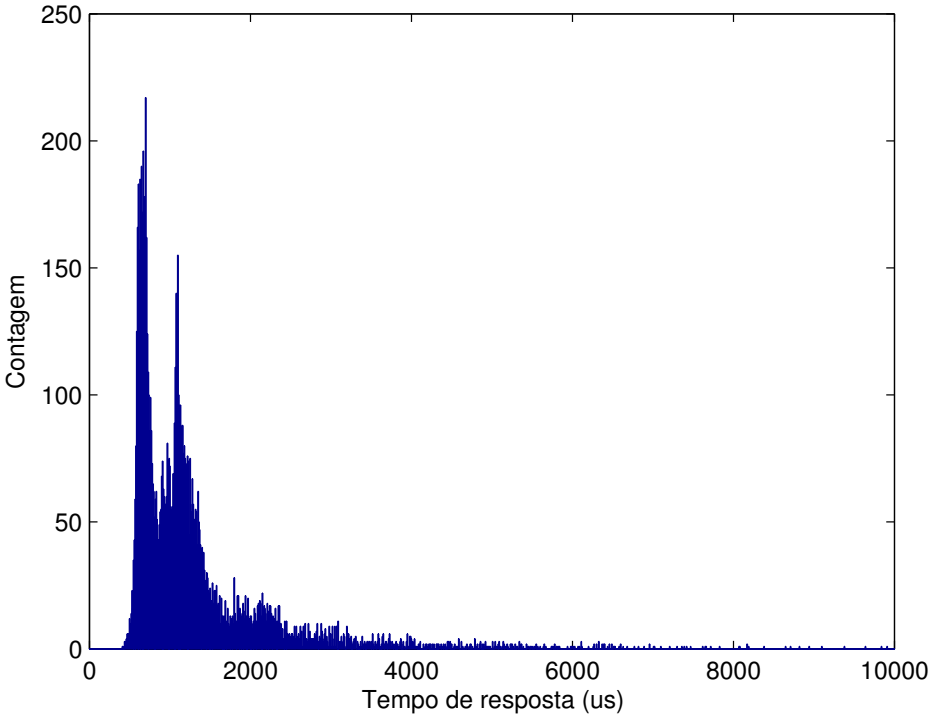

Figura 5: Histograma de Respostas Temperatura em Alta Atividade

2.2.2 Distribuição Normal de Probabilidade

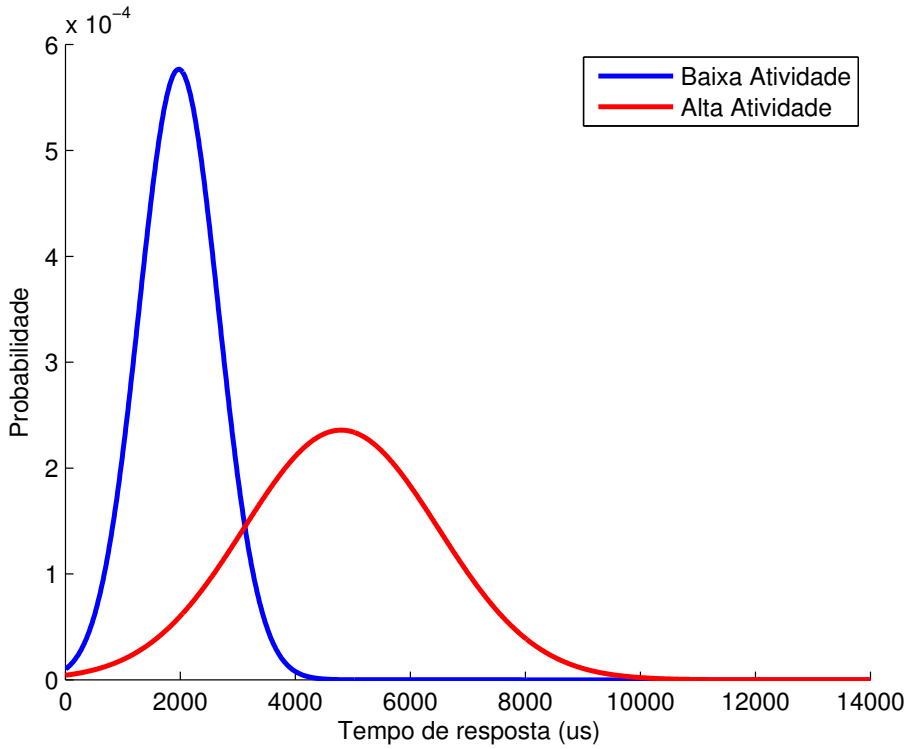

Figura 6: Função Densidade de Probabilidade Temperatura