9
Estudo de Caso

9.1
Domínio

Escolhemos como estudo de caso aplicar o McCloud no apoio de uma pesquisa da área de Engenharia Mecânica. Tradicionalmente, os projetos desta área (automobilística, construção, mineração, petróleo e gás, etc.) utilizam um fator de segurança para compensar imprecisões nos modelos preditivos utilizados. Essas imprecisões são provenientes de imperfeições nas dimensões físicas do sistema/objeto em análise, da variabilidade nominal das propriedades físicas ou por diversa outras razões, e na maioria das vezes, são originadas por pura falta de conhecimento dos fenômenos de interesse.

Esses fatores tem se revelado uma solução eficiente para evitar cenários de falha, mas podem comprometer a viabilidade financeira do projeto se não forem bem estimados. Uma alternativa ao cálculo determinístico do fator de segurança, para estimação mais precisa, é a utilização da Simulação de Monte Carlo com altos números de realizações, para definir um intervalo de confiabilidade, onde a probabilidade de falha é menor que um determinado valor predefinido.

O exemplo adotado [26] estuda a quantidade física referente ao deslocamento de uma barra elástica unidimensional, que tem uma densidade de massa constante, área de secção transversal circular constante e um dado comprimento. A elasticidade da barra é o parâmetro aleatório deste sistema contínuo.

O sistema teórico proposto no estudo consiste em fixar o lado esquerdo da barra a uma parede rígida, enquanto, o lado direito é ligado a uma mola fixa em outra parede. Essa mola é linear e exerce uma força de restauração proporcional ao alongamento na barra. Na barra atua também uma força externa, que depende da posição e do tempo. A quantidade física de interesse, deslocamento da barra, é uma função da posição e do tempo. A ilustração do sistema, nomeado de “Sistema 0”, é apresentado a seguir.

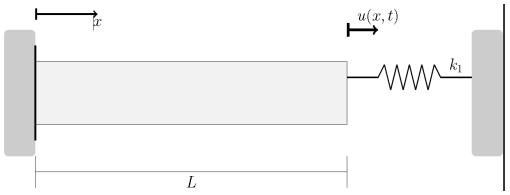

Figura 35 – Sistema 0 do Estudo de Caso

Posteriormente, para aumentar a complexidade do sistema, é adicionada uma segunda mola não linear, cuja força de restauração é proporcional ao cubo do alongamento, fixando também a extremidade direita da barra à parede rígida. A ilustração do sistema com a adição dessa mola, nomeado de "Sistema 1", é apresentada a seguir.

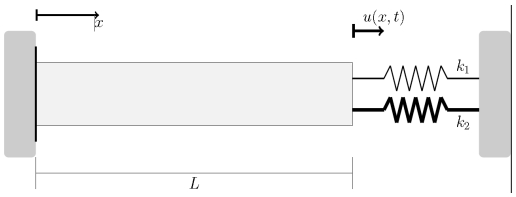

Figura 36 – Sistema 1 do Estudo de Caso

Sem entrar no mérito da modelagem proposta do estudo, como curiosidade, a variável aleatória do sistema é a elasticidade da barra, aderente a Distribuição Gama [45], e os resultados determinísticos são obtidos com o Método de Galerkin [44]. Dessa forma, os elementos necessários para realização da simulação de Monte Carlo estão definidos e se podem calcular as estatísticas do deslocamento em cada sistema.

9.2
Cenário Atual

O algoritmo da Simulação de Monte Carlo, proposto neste estudo, foi desenvolvido pelos pesquisadores em MATLAB [27], segundo o fornecedor, uma linguagem de programação apropriada a problemas de natureza técnica, que possui facilidades de computação, visualização e programação, dentro de um ambiente amigável e de fácil aprendizado.

O MATLAB foi originalmente desenvolvido para prover um acesso amigável ao tratamento de vetores e matrizes. No entanto, atualmente dispõe de uma biblioteca bastante abrangente de funções matemáticas, geração de gráficos e manipulação de dados.

A execução desse algoritmo em uma única máquina é extremamente lenta, não permitindo a geração de uma massa de resultados suficiente para comprovar que o modelo proposto está correto, ou apontar falhas. Atualmente o máximo de realizações passíveis de serem processadas na infraestrutura existente é de 1.024, pois em simulações com um número maior de realizações ocorre lotação da memória. No patamar máximo a execução leva em torno de 90 minutos na máquina do pesquisador, considerando o sistema mais complexo (Sistema 1).

O investimento em equipamentos para montar um *cluster* local para processamento deste algoritmo representaria um investimento muito grande frente aos recursos disponíveis para pesquisa, principalmente, pelo fato de não se precisar desta infraestrutura após a finalização da pesquisa. Tal estratégia demandaria necessidade de licenças específicas para execução em paralelo do MATLAB. Além disso, ainda enfrentaríamos riscos de lotação da memória.

Diante do cenário exposto, os pesquisadores envolvidos despertaram o interesse pela possibilidade de utilizar o McCloud para viabilizar a execução deste algoritmo de forma fácil e eficiente no ambiente de computação na nuvem. O objetivo prático era alcançar uma aproximação se utilizando de 262.144 realizações do Sistema 1.

9.3 Adaptações

A primeira providência foi dividir o código MATLAB em duas partes, assim como feito na prova de conceito, uma para realização dos passos B e C da simulação de Monte Carlo (seção 2.2), ou seja, geração das variáveis pseudo-aleatórios e cálculo determinístico, e a segunda parte para realização do passo D (seção 2.2), obtenção das estatísticas.

Para evitar problemas de memória, a entrada e saída de dados de cada parte, passou a ser realizada com arquivos de texto, utilizando-se as funções *fopen*, *fprintf* e *fscanf* do MATLAB. Além disso, para evitar problemas de repetição de números pseudo-aleatórios colocamos um índice para variação da semente do seu gerador.

Com o MATLAB compilamos com a função *deploytool* essas duas partes criando dois executáveis *standalone*, cuja chamada apresentamos a seguir.

```
>> process n s flag output
```

O *n* representa o total de realizações (amostra) a ser processado nessa execução, o *s* é o índice a ser utilizado no gerador de números pseudo-aleatórios, o *flag* indica o número do sistema a ser adotado (Sistema 0 ou Sistema 1) e o *output* indica o caminho e nome do arquivo a ser salvo com os dados de saída do executável.

```
>> merge n input output
```

O *n* representa o total de realizações (amostra), o *input* indica o caminho e nome do arquivo de entrada com o resultado concatenado de todas as execuções do executável anterior e o *output* indica o caminho e nome do arquivo a ser salvo com os dados de saída deste executável.

Cabe observar que todas essas entradas são textuais, portanto, devem ser alimentadas entre cotas ('texto') e dentro do MATLAB utilizar a função *str2num* quando for necessário converter de texto para número.

Ao executarmos o primeiro executável para 1.024 realizações, tivemos uma duração praticamente igual à encontrada antes das modificações, ou seja, não ocorreu um acréscimo significativo de esforço. No entanto, percebemos que o primeiro executável passava um volume muito grande de dados para o segundo. A projeção deste número para nosso objetivo de 262.144 realizações estaria na ordem de 200GB. Portanto, passamos a analisar uma forma de armazenar menos informações, visto que estamos interessados unicamente nas estatísticas finais.

As estatísticas de interesse são média, variância e a função de densidade de probabilidade (PDF), da extremidade direita da barra, sendo que as duas primeiras são obtidas em função do tempo e a última para um instante fixado. Dessa forma, passamos, primeiramente, a armazenar apenas os dados referentes à extremidade direita da barra e não mais de diversos outros pontos da barra, para obtenção da PDF. Além disso, adotamos técnicas de aproximação da média e da variância para permitir acumular poucos dados em cada tarefa [43].

O algoritmo da média apresentado em E13 permite aproximar a média da amostra a partir do tamanho (*n*) e da média (μ) de duas partes da mesma (*a* e *b*).

$$\text{(E13)}\quad \mu_{ab} = \mu_a + \frac{n_b(\mu_b - \mu_a)}{n_a + n_b}$$

O algoritmo da variância apresentado em E14 permite aproximar a variância da amostra a partir do M e do tamanho (*n*) de duas partes da mesma (*a* e *b*), onde M é obtido através da soma ao quadrado das diferenças de todos os pontos com a média.

$$M = \sum_1^n (x_i - \mu)^2$$

(E14)

$$\sigma^2 = M_a + M_b + \frac{n_b n_a (\mu_b - \mu_a)}{n_a + n_b}$$

Com essa abordagem passamos a armazenar para cada tarefa apenas a média (μ), a soma ao quadrado das diferenças de cada ponto com a média (M) e o deslocamento da extremidade direita da barra no último instante de tempo (ULT). Dessa forma, a projeção dos dados armazenados ficou menor que 1GB para essa mesma amostra.

Ainda sem utilizar o McCloud ou a computação na nuvem, lançando mão apenas destas modificações feitas dentro de uma visão de paralelismo, torna-se possível chegar ao número de realizações desejado da seguinte forma:

a. Executando 256 vezes o primeiro executável com 1.024 realizações (*n*) cada e incrementando o índice (*s*) em cada execução de forma a não repetir a geração dos números pseudo-aleatórios, pois como demonstrado anteriormente, uma execução única estoura a memória;
b. Concatenar os 256 arquivos resultados em um único arquivo, ou seja, copiando e colando no final do primeiro cada arquivo gerado;
c. Passando então, a executar o segundo executável com a amostra concatenada.

Somente o primeiro passo, não levaria menos de 23.040 minutos (90 minutos x 256 repetições), ou seja, 16 dias.

9.4 Implementação

Como o algoritmo em questão possui significativa complexidade, e possivelmente precisará ser evoluído de forma a acompanhar as evoluções no modelo proposto, objeto da pesquisa, é injustificável reescrever esse algoritmo MATLAB em C#.Net para adaptá-lo a solução genérica disponibilizada na prova de conceito apresentada no capítulo 5. Além disso, os pesquisadores gostariam de

continuar utilizando a linguagem MATLAB, possuem bom conhecimento e são capazes de promover alterações livremente.

Dessa forma, adotamos a estratégia de utilizar os executáveis gerados pelo MATLAB, e adaptados na seção anterior, nos pontos de extensão do McCloud, conforme Figura 38. Para isso, se torna necessário instalar nas instâncias o MCRInstaler (*Matlab Compiler Runtime*), requisito para uso destes executáveis. Isto foi feito estendendo o ponto de extensão *startup*, conforme Figura 37.

Repare que o McCloud disponibiliza o *"BootStrapper"* [28] para gerenciamento de *downloads* e instalações, e que utilizamos um artifício para criar um usuário administrativo e executar a instalação como este, visto que o nó não possui usuário. Além disso, carregamos os executáveis MATLAB no blob "onstart" para que ele seja copiado automaticamente para a pasta de armazenamento local das instâncias.

Esses executáveis (*standalone*) foram nomeamos de *process* e *merge*. Os mesmos ficarão acessíveis na pasta local chamada "McCloudStorage" [29][30][31].

```
@echo off
SETLOCAL
SET regpath=HKLM\Software\McCloud

REG QUERY %regpath% /v vcredist_x86
IF errorlevel 1 (
 BootStrapper.exe -get http://mccloud1.blob.core.windows.net/startup/vcredist_x86.exe -lr C:\mccloud\ -run C:\mccloud\vcredist_x86.exe -args /w /s /v/qn -block
 REG ADD %regpath% /v vcredist_x86 /t REG_SZ /d 1 /f
)

REG QUERY %regpath% /v MCRInstaller
IF errorlevel 1 (
 BootStrapper.exe -get http://mccloud1.blob.core.windows.net/startup/MCRInstaller.exe -lr C:\mccloud\ -block

 net start "task scheduler"
 net user McCloud m4c2d4h49 /add
 net localgroup Administrators McCloud /add

 schtasks /Create /SC ONCE /ST 00:00 /SD 01/01/2100 /TN task_MCRInstaller /TR "C:\mccloud\MCRInstaller.exe /w /s /v/qn" /F /RU McCloud /RP m4c2d4h49 /RL HIGHEST
 schtasks /Run /TN task_MCRInstaller

 REG ADD %regpath% /v MCRInstaller /t REG_SZ /d 1 /f
 shutdown -r -t 390
)
:EXIT
exit /b 0
```

Figura 37 – Ponto de extensão startup do Deslocamento da Barra

A implantação proposta dos pontos de extensão é apresentada nas figuras a seguir. Repare que os executáveis são chamados pelo nome de *process* e *merge* da pasta local, onde foram gravados na inicialização de cada instância.

```
public class RandBar : McHotspootI
{
    public string execute(string key, double n, double index, string codein, out string message)
    {
        string filename = "process"; double myN = 1;
        string filesRoot = RoleEnvironment.GetLocalResource("McCloudStorage").RootPath;
        string matlab1 = Path.Combine(filesRoot, filename + ".exe");
        string output1 = "P_" + key + index + ".csv";

        exec(matlab1, new string[] {myN.ToString(), index.ToString(), codein, output1}, out message);
        FileStream fileStream = new FileStream(output1, FileMode.Open);
        StreamReader reader = new StreamReader(fileStream);
        string r = reader.ReadToEnd();
        fileStream.Close();
        return r;

    }
    public string finish(string key, double n, string r, string codein, string codeout,
        out string message)
    {
        string filename = "merge"; double myN = ((Math.Log(n, Math.E) / Math.Log(4, Math.E))-3);
        string filesRoot = RoleEnvironment.GetLocalResource("McCloudStorage").RootPath;
        string matlab2 = Path.Combine(filesRoot, filename + ".exe");
        string output1 = r;
        string output2 = "M_" + key + ".csv";
        exec(matlab2, new string[] { myN.ToString(), output1, output2 }, out message);
        string m = "";
        FileStream fileStream2 = new FileStream(output2, FileMode.OpenOrCreate);
        StreamReader reader = new StreamReader(fileStream2);
        m = reader.ReadToEnd();
        fileStream2.Close();
        return m;
    }
    public void optimization(double n, string codein, string codeout,
        out double ninstancesadded, out double ntasks, out int timeoutInSeconds)
    {
        double nTasks = n / 256;
        ntasks = Decimal.ToInt32(Decimal.Truncate((decimal)(nTasks)));
        ninstancesadded = 0;
        timeoutInSeconds = 2 * 60 * 60; // Máximo 2 horas
    }
    public string test(string key, double n, string codein, string codeout)
    {
        if (codein != "1" && codein != "0") return "Error: Incorrect Imput";
        return "";
    }
    private void exec(string file, string[] arguments, out string message)
    {
        if (!System.IO.File.Exists(file))
        {
            message = "Executável inexistente: " + file;
            return;
        }
        System.Diagnostics.ProcessStartInfo psi = new System.Diagnostics.ProcessStartInfo();
        psi.FileName = file;
        psi.Arguments = "";
        for (int i = 0; i < arguments.Length; i++){ psi.Arguments += " \"" + arguments[i] + "\""; }
        psi.Arguments = string.Format(psi.Arguments);
        psi.CreateNoWindow = true;
        psi.ErrorDialog = false;
        psi.UseShellExecute = false;
        psi.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
        psi.RedirectStandardOutput = true;
        psi.RedirectStandardInput = false;
        psi.RedirectStandardError = true;

        System.Diagnostics.Process exeProcess = System.Diagnostics.Process.Start(psi);
        exeProcess.PriorityClass = System.Diagnostics.ProcessPriorityClass.High;
        exeProcess.OutputDataReceived +=
            new System.Diagnostics.DataReceivedEventHandler(processOutputHandler);
        exeProcess.BeginOutputReadLine();
        string errString = exeProcess.StandardError.ReadToEnd();
        exeProcess.WaitForExit();
        exeProcess.Close();
        message = "Exec: " + psi.FileName + " " + psi.Arguments + "\r\n";
        if (errString != "") message += "Exec Erro: " + errString;
        if (_output != "") message += "Exec Saída: " + _output;
    }
```

Figura 38 – Pontos de extensão do Deslocamento da Barra (parte 1)

```
    private string _output = "";
    private void processOutputHandler(object sendingProcess,
        System.Diagnostics.DataReceivedEventArgs outLine)
    {
        if (!String.IsNullOrEmpty(outLine.Data)) _output += outLine.Data+"\r\n";
    }
}
```

Figura 39 – Pontos de extensão do Deslocamento da Barra (parte 2)

Repare que o *codein* é o parâmetro 0 ou 1 para indicar o sistema da simulação, o *codeout* não é utilizado, o *test* é apenas para verificar o parâmetro de entrada e o *optimazation* não alterará a quantidade de nós, que será fixa durante toda a execução. Os executáveis são chamados dos pontos de extensão *execute* e *finish* e seus resultados em arquivo são manipulados mantendo o funcionamento aderente a interface. Cabe destacar que eventuais necessidades de passagem de outros parâmetros para o primeiro executável, podem ser realizadas utilizando algum separador no texto do *codein*.

Para melhor representação visual dos dados estatísticos finais, sugere-se que o número de realizações (*n*) seja uma potência de 4. Sendo assim, e com foco em melhor acompanhamento da execução e possibilidade de comparação da porção, dividimos a simulação em tarefas de 256 realizações cada.

O cliente deste serviço foi desenvolvido também em PHP5, mas desta vez com uma interface com uma pequena lógica, a saber:

a) A primeira página apresenta um formulário para escolha da ordem da amostra e do número do sistema a ser simulado, conforme abaixo.

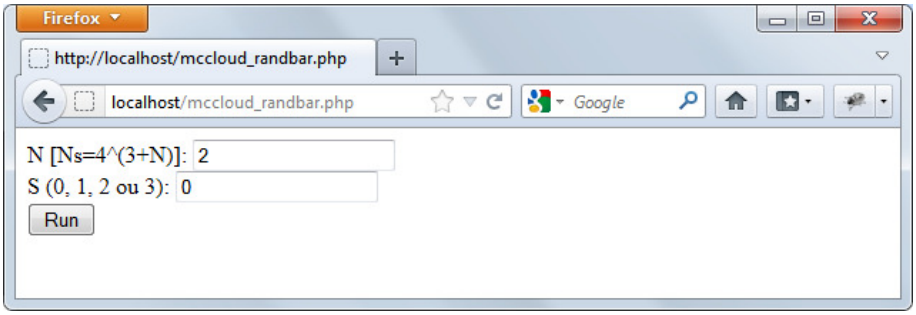

Figura 40 – Página de Início da Simulação do Estudo de Caso

b) A segunda página, que é exibida ao clicar em “Run” na página acima, permite checar o andamento desta simulação, conforme abaixo. O status varia de acordo com a situação atual da simulação e pode ser atualizado clicando em “Check”.

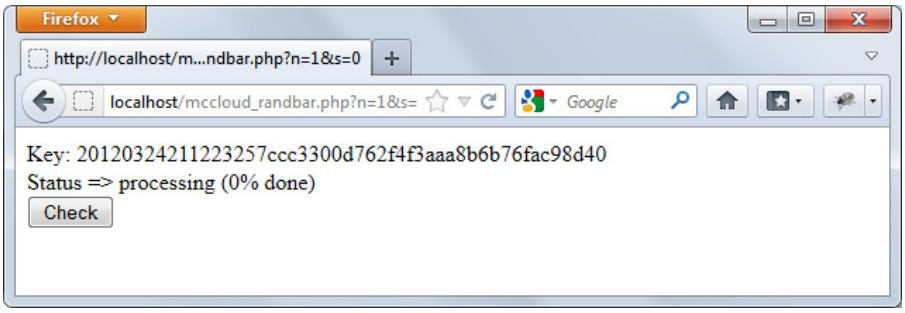

Figura 41 – Página de Checagem da Simulação do Estudo de Caso

a) A terceira e última página aparece quando a simulação finalizou e se clica em “Check” na página anterior. O botão para *download* dos resultados desta simulação é apresentado.

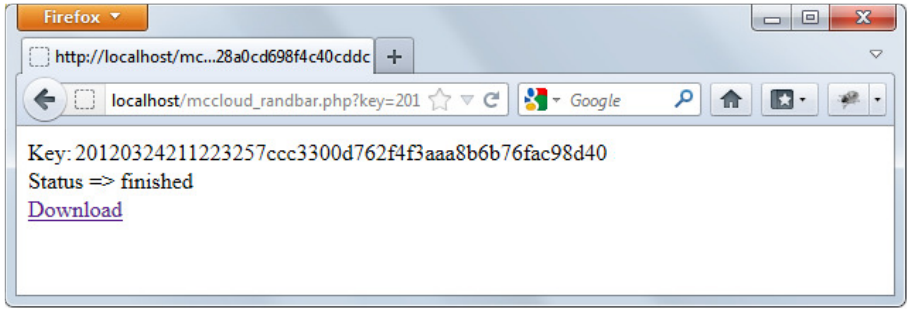

Figura 42 – Página de Download da Simulação do Estudo de Caso

Apresentamos a seguir o código em PHP5 responsável por esta aplicação.

```php
<?php
set_time_limit (0);
$end = 'http://mccloudservice2.cloudapp.net';
$wsdl = $end.'/Service.svc?wsdl';
$mcc = new SoapClient($wsdl, array("connection_timeout"=>10000000));

if(!empty($_GET['r'])){
	$obj4->key = $_GET['key'];
	$result4 = $mcc->Merge($obj4);
	$ok = $result4->ResultResult;
	echo $ok;
	die();
}

if(empty($_GET['n']) && empty($_GET['key'])){
	?>
	<form id="frm1" name="frm1" method="get" action="">
		N [Ns=4^(3+N)]: <input name="n" id="n" type="text" value="2"/><br>
		S (0, 1, 2 ou 3): <input name="s" id="s" type="text" value="0"/><br>
		<input type="submit" value="Run"/>
	</form>
	<?
}
if(!empty($_GET['key'])) $key = $_GET['key'];
if(!empty($_GET['n']) && empty($_GET['key'])){
	$obj->n =  Pow(4,(3+$_GET['n']));
	$obj->codein = $_GET['s'];
	$obj->codeout = '';

	$result = $mcc->Run($obj);
	$key = $result->RunResult;
}

if(!empty($key)){
	$obj2->key = $key;
	$result2 = $mcc->Check($obj2);
	$status = $result2->CheckResult;
	echo "Key: ".$key."<br>";
	echo "Status => ".$status."<br>";
	if($status != "finished"){
		?>
		<form id="frm2" name="frm2" method="get" action="">
			<input name="key" id="key" type="hidden" value="<?=$key?>"/>
			<input type="submit" value="Check"/>
		</form>
		<?
	} else {
		$obj3->key = $key;
		$result3 = $mcc->Result($obj3);
		$url = $result3->ResultResult;
		?><a href="<?=$url?>" target="_blank">Download</a><?
	}
}
?>
```

Figura 43 – Aplicação Cliente Deslocamento da Barra

9.5 Performance

Apresentamos na Tabela 8 a seguir o resultado da simulação com diferentes valores para *n* (número de realizações), *w* (número de instâncias trabalhadoras), sistema (0 ou 1) e ambientes. O ambiente "nuvem" refere-se à execução na plataforma Microsoft Windows Azure, enquanto no ambiente "emulador" ocorre a

execução na máquina local de desenvolvimento através de ferramentas que emulam o Azure. No ambiente "*standalone*" não é utilizado o McCloud ou emulador ou a nuvem, sendo apenas rodados os executáveis na mesma máquina local.

RAND BAR PERFORMANCE															
Entrada				Configuração			Tempo (minutos)				Blob		CUSTO	Comparação	
#	n	Sistema	Ambiente	W	Tarefas	Tar./W	Split	Process	Merge	Total	Interno	Saída	USD$	Tempo	Razão
1	256	0	Standalone	-	-	-	0,00	2,13	0,05	**2,17**	882KB	879KB	-	-	-
1	256	1	Standalone	-	-	-	0,00	11,72	0,04	**11,76**	882KB	879KB	-	-	-
1	256	0	Emulator	-	-	1	0,02	2,51	0,14	**2,67**	882KB	879KB	-	2	-
2	1.024	0	Emulator	-	-	4	0,00	7,94	0,15	**8,09**	3,5MB	879KB	-	9	-
1	256	1	Emulator	-	-	1	0,00	11,34	0,13	**11,47**	882KB	879KB	-	12	-
2	1.024	1	Emulator	-	-	4	0,03	45,43	0,17	**45,62**	3,5MB	879KB	-	47	-
1	256	0	Cloud	1	1	1	0,02	2,43	0,16	**2,61**	882KB	879KB	**0,39**	2	0,83
2	1.024	0	Cloud	4	4	1	0,01	2,53	0,20	**2,73**	3,5MB	879KB	**0,75**	9	3,18
3	4.096	0	Cloud	16	16	1	0,02	2,72	0,41	**3,15**	13,8MB	879KB	**2,19**	35	11,03
4	16.384	0	Cloud	19	64	4	0,08	9,44	1,17	**10,69**	55,1MB	879KB	**2,55**	139	13,01
5	65.536	0	Cloud	19	256	14	0,10	32,84	3,06	**36,00**	220,5MB	879KB	**2,55**	556	15,45
6	262.144	0	Cloud	19	1024	54	0,34	134,96	8,06	**143,36**	882MB	879KB	**7,35**	2.225	15,52
1	256	1	Cloud	1	1	1	0,01	11,63	0,13	**11,77**	882KB	879KB	**0,39**	12	1,00
2	1.024	1	Cloud	4	4	1	2,08	12,08	0,17	**14,33**	3,5MB	879KB	**0,75**	47	3,28
3	4.096	1	Cloud	16	16	1	0,02	12,61	0,25	**12,88**	13,8MB	879KB	**2,19**	188	14,62
4	16.384	1	Cloud	19	64	4	0,10	48,20	1,24	**49,54**	55,1MB	879KB	**2,55**	753	15,20
5	65.536	1	Cloud	19	256	14	0,35	165,81	5,38	**171,54**	220,5MB	879KB	**7,35**	3.012	17,56
6	262.144	1	Cloud	19	1024	54	0,34	643,48	7,96	**651,78**	882MB	879KB	**26,55**	12.047	18,48

Tabela 8 – Teste de performance do estudo de caso (sistema 0 e 1)

É evidente que a solução para um número baixo de realizações não faz sentido, conforme anteriormente observado na prova de conceito. Ainda mais neste Estudo de Caso, devido à necessidade de download e instalação do *Matlab Compiler Runtime*, que obriga a reinicialização da instância após a instalação, o que aumenta o tempo de ativar os nós e posteriormente desativá-lo para girou em torno de 20 minutos. Esse tempo não é considerado na tabela acima.

É interessante comentar que poderíamos adotar outra estratégia em vez de instalar os requisitos do MATLAB na inicialização de cada instância, por exemplo, espelhando a imagem de uma máquina modelo, no entanto, essas abordagens demandariam do usuário um conhecimento muito profundo, que não é interessante para a popularização da solução.

Considerando o objetivo dos pesquisadores, alcançar uma amostra com 262.144 realizações do Sistema 1, temos na tabela acima, com 19 instâncias (w), a duração (T) de menos de 11 minutos e custo (C) de USD$ 26,55. Em relação ao tempo linear (*Comparação>Tempo*), apresentado nesta mesma linha da tabela, o ganho foi na ordem de 18 vezes mais rápido (*Comparação>Razão*).

A proporcionalidade observada em todos os experimentos do estudo de caso em questão, da razão do ganho (*Comparação>Razão*) com o número de nós (W), nos indica que o tempo de gerenciamento não onera de forma significativa a eficiência do experimento.

Apresentamos a seguir as funções de densidade de probabilidade (FDPs) do Sistema 1. As FDPs da primeira linha são da computação tradicional com 256 (a) e 1.024 (b) realizações. As FDPs da segunda e terceira linha são da implementação do McCloud com 4.096 realizações (c), 16.384 realizações (d), 65.536 realizações (e) e 262.144 realizações (f).

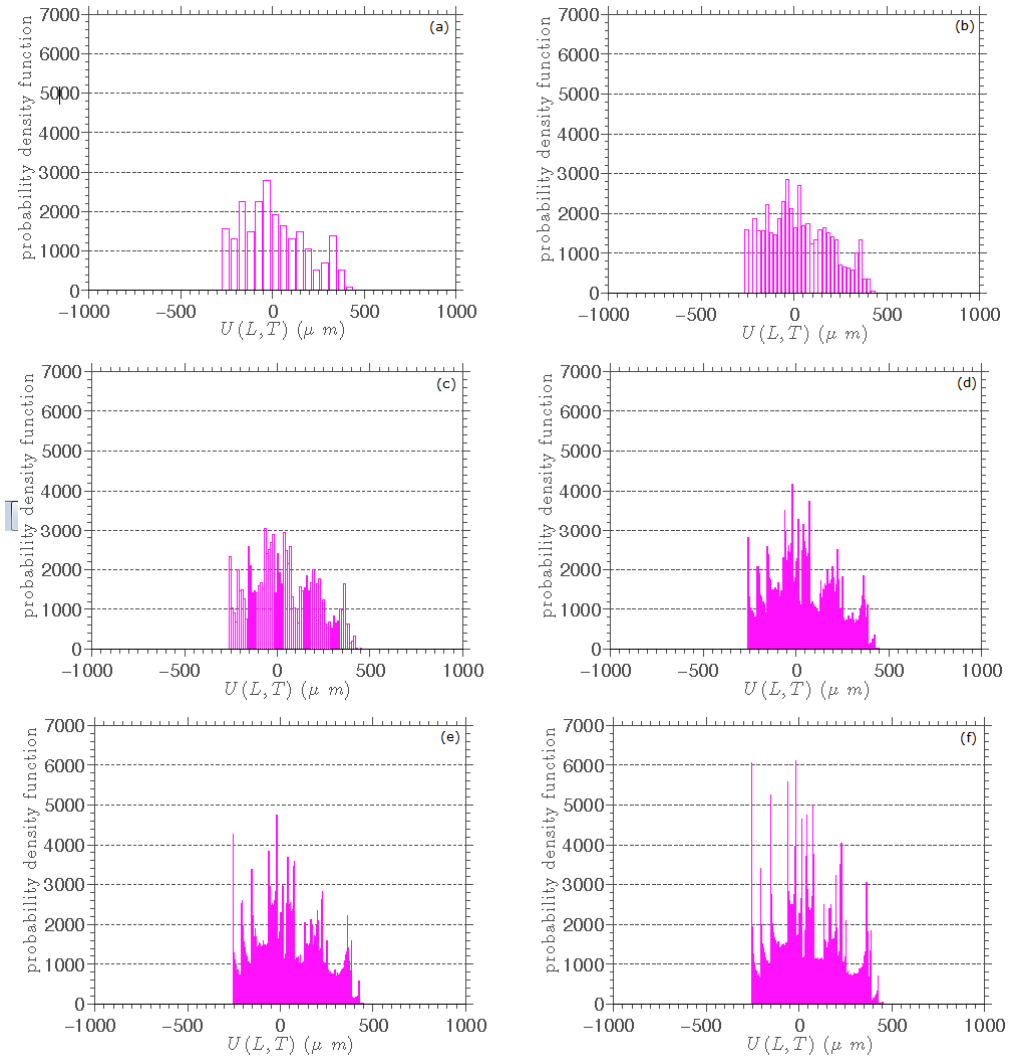

Figura 44 – FDP do sistema 1 do estudo de caso

A opinião dos pesquisadores em relação a essas simulações foi *"Em termos estatísticos, sem dúvida houve benefícios em todos os sistemas. A função densidade de probabilidade muda bastante quando passamos de uma amostra de*

1.024 para 265.144. Entretanto, devido à simplicidade do Sistema 0 e 1, por exemplo, teríamos uma boa estatística com 16.384 realizações.”. Essa conclusão se torna possível somente ao realizar a amostra de 65.536 realizações e identificar que o comportamento se mantém. Dessa forma, a amostra de 262.144 realizações, nesse contexto, foi um preciosismo.

Diante da análise dos resultados obtidos, os pesquisadores inseriram dois novos elementos no sistema, aumentando sua complexidade, um fluido viscoso em torno da barra e uma massa na extremidade direita da barra, criando assim, um amortecimento e tornando a dinâmica do sistema mais lenta. Considerando as duas variações anteriores, com ou sem a segunda mola, obtivemos dois novos sistemas com estes elementos, nomeados de “Sistema 2” e “Sistema 3”, conforme ilustrado abaixo.

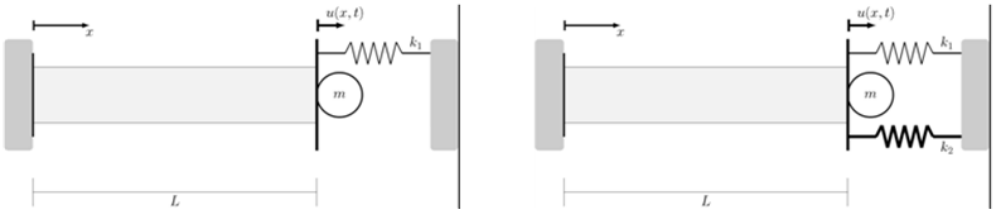

Figura 45 – Sistema 2 (esquerda) e 3 (direita) do Estudo de Caso

Nesses novos sistemas, tornou-se necessário estudar a dinâmica da barra por mais, visto que se tornou mais lenta. Com isso, cresceu o volume de informações a serem armazenados como resultado de cada tarefa.

Os novos experimentos realizados com o Sistema 2 e Sistema 3 são apresentados a seguir. Repare que reduzimos o número de experimentos para os de maior relevância para essa segunda análise.

RAND BAR PERFORMANCE																
Entrada				Configuração			Tempo (minutos)				Blob		CUSTO	Comparação	
#	n	Tipo	Ambiente	W	Tarefas	Tar./W	Split	Process	Merge	Total	Interno	Saída	USD$	Tempo	Razão
1	256	2	Standalone	-	-	-	0,00	2,44	0,05	**2,49**	1,3MB	1,4MB	-	-	-
1	256	3	Standalone	-	-	-	0,00	10,96	0,05	**11,01**	1,3MB	1,4MB	-	-	-
1	256	3	Emulator	-	-	1	0,01	11,50	0,16	**11,67**	1,3MB	1,4MB	-	11	-
1	256	3	Cloud	1	1	1	0,00	11,37	0,14	**11,51**	1,3MB	1,4MB	**0,39**	11	0,96
2	1.024	3	Cloud	4	4	1	0,01	11,87	0,17	**12,05**	5,2MB	1,4MB	**0,75**	44	3,65
3	4.096	3	Cloud	16	16	1	0,01	12,15	0,32	**12,48**	20,8MB	1,4MB	**2,19**	176	14,11
4	16.384	3	Cloud	19	64	4	0,03	46,30	0,27	**46,61**	83,2MB	1,4MB	**2,55**	705	15,12
5	65.536	3	Cloud	19	256	14	0,09	161,98	3,14	**165,21**	332,8MB	1,4MB	**7,35**	2.818	17,06
6	262.144	3	Cloud	19	1024	54	1,50	638,71	28,27	**668,47**	1,3GB	1,4MB	**28,95**	11.273	16,86

Tabela 9 – Teste de performance do estudo de caso (sistema 2 e 3)

Considerando o objetivo dos pesquisadores, alcançar uma amostra com 262.144 realizações do Sistema 3, temos na tabela, com 19 instâncias (w), a

duração (*T*) de menos de 11,14 horas e custo (*C*) de USD$ 28,95. Em relação ao tempo linear (*Comparação>Tempo*), apresentado nesta mesma linha da tabela, o ganho foi na ordem de 16 vezes mais rápido (*Comparação>Razão*).

Apresentamos a seguir as funções de densidade de probabilidade (FDPs) do Sistema 3, com 256 (a), 1.024 (b) 4.096 (c), 16.384 (d), 65.536 (e) e 262.144 (f) realizações.

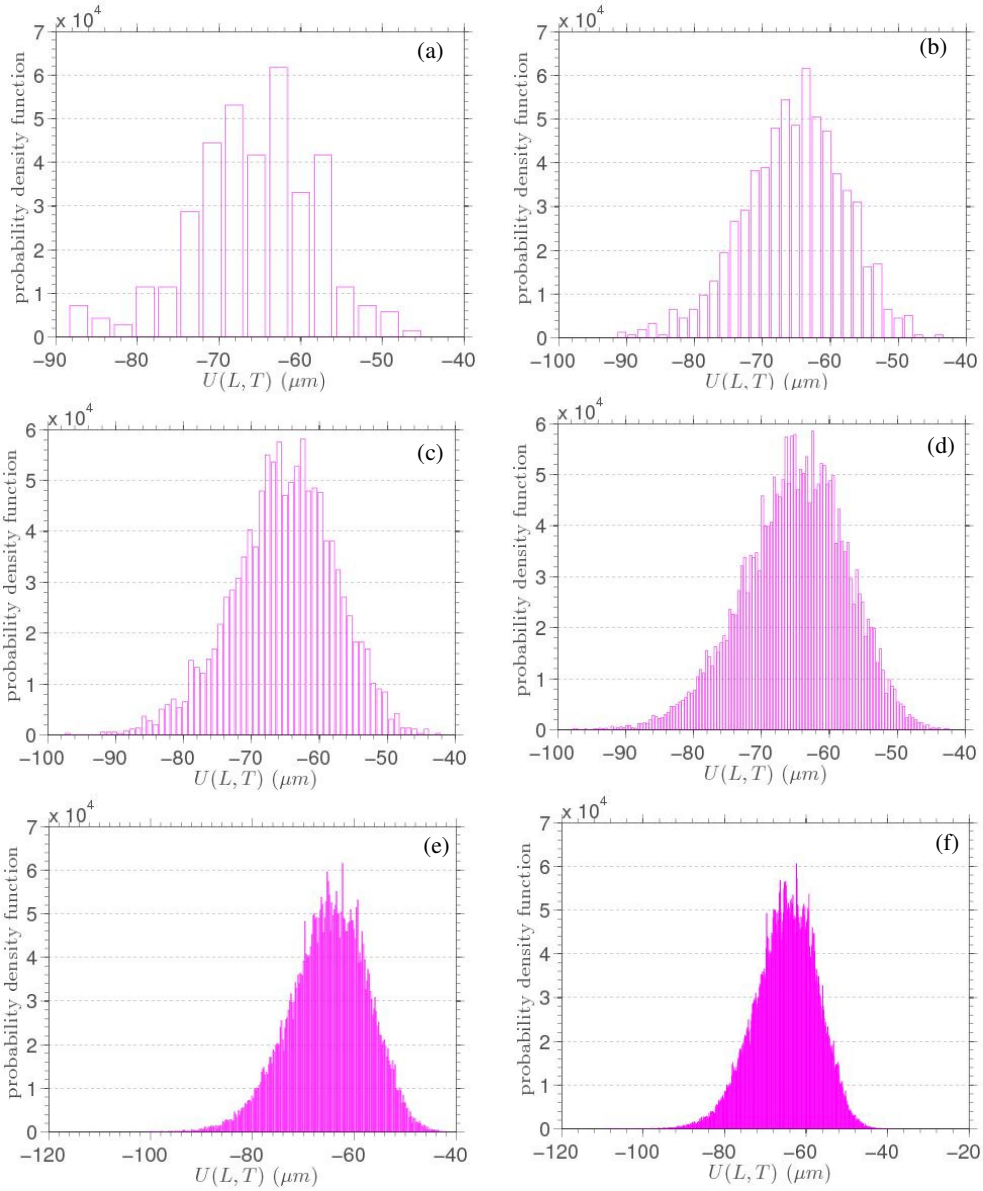

Figura 46 – FDP do sistema 3 do estudo de caso

Na Figura 47, a seguir, apresentamos a média (linha azul no centro) e a região relativa a dois desvios padrões acima e abaixo da média (em amarelo) para

1.024 (a) e 262.144 (b) realizações do Sistema 3. A redução da região ilustra o ganho de precisão.

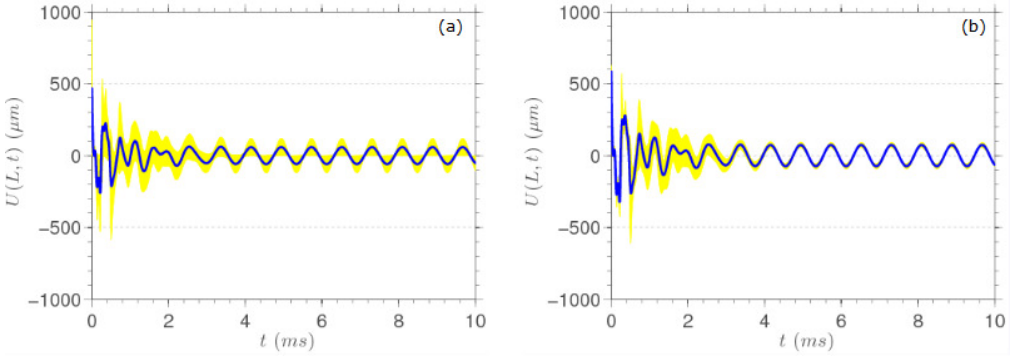

Figura 47 – Média do sistema 3 do estudo de caso

A opinião dos pesquisadores em relação a essas simulações foi *"Como previsto, dada a maior complexidade do Sistema 2 e 3, tornou-se necessário as 265.144 realizações para análise adequada do sistema."*.

9.6
Análise dos Resultados

Conforme descrito anteriormente no capítulo 7, serviços implementados com o McCloud em ambientes de computação na nuvem permitem simulações mais rápidas e custo proporcional aos requisitos do experimento, sem ociosidade ou escassez de recursos.

Os resultados obtidos com um número maior de realizações, conforme é possível visualizar nas funções de densidade de probabilidade apresentadas, permitem uma análise mais apurada e precisa do comportamento da incerteza em estudo. Portanto, é evidente que a solução proposta permite novas perspectivas para aplicação da simulação de Monte Carlo e pode apoiar diversas pesquisas.

Um dos pesquisadores, Ph.D., com mais de 30 anos de experiência, observou: *"Nunca realizei em toda minha carreira uma simulação com mais de 16.000 realizações"*.

Quanto à implantação realizada nesse estudo de caso com o McCloud, repare que os executáveis MATLAB são carregados em cada inicialização do serviço a partir do container "onstart" e chamados pelos nomes de *process* e *merge*. Com isso, essa implementação, sem qualquer alteração, pode ser utilizada para execução de outras simulações desenvolvidas no MATLAB. Para isso, basta

adaptar o código MATLAB aos conceitos discutidos e interfaces adotadas na seção 9.3 e, após compilá-los como executáveis *standalone* e nomeá-los como *process* e *merge*, enviar para esse container. Além disso, é possível flexibilizar a passagem de parâmetros de entrada, passando-os pelo *codein* com separadores textuais, a serem tratados internamente no MATLAB.

Esse estudo de caso representa o potencial que a solução proposta tem para alavancar o volume de simulações adotadas em pesquisas com orçamentos reduzidos.