Analyzing the Impact of Power Subsystem Failures and Checkpoint Mechanisms on Availability of Cloud Applications

É. Rocha, G. Leoni, and P.Takako

***Abstract*—The widespread adoption of cloud computing has been driven by technological advantages, such as pay-as-you-go plan, scalability and high availability. From cloud service providers point-of-view, managing all these high customer expectations is a growing and critical challenge. This paper examines the relationship among energy infrastructure architectures, checkpoint mechanisms, and availability of software applications in a cloud data center. We consider four power architectures, three checkpoint mechanisms (cold, warm, and hot) with different sizes of files, and two applications (Video on Demand and digital library). For that, we propose a set of stochastic models (based on Petri Net and Reliability Block Diagram) and perform sensitivity and availability analysis. From results, we note that the power architecture has a great impact on application availability while the checkpoint mechanism only impacts with the application has a low mean time to failure value.**

***Index Terms*—Power Subsystem, Checkpoint mechanisms, Cloud applications.**

I. INTRODUÇÃO

DADA a capacidade computacional limitada dos dispositivos localizados na borda da Internet, como *smartphones*, *tablets* e dispositivos *wearables*, atualmente é possível transferir parte do processamento realizado nesses dispositivos para o ambiente de nuvem computacional. De acordo com o NIST (*National Institute of Standards and Technology*) [1], computação em nuvem pode ser definido como "*um modelo computacional que permite o acesso a dados e informações dispontíveis na rede de maneira ubíqua*". Assim, pode-se identificar como ponto principal dos ambientes em nuvem, o compartilhamento e a integração de recursos distribuídos para o aumento e disponibilidade da capacidade de processamento de uma determinada aplicação [2].

Atualmente, os provedores de nuvem utilizam *data centers* para hospedar as aplicações de seus clientes. Porém, estes *data centers* possuem uma infraestrutura bastante complexa, sendo composta por três subsistemas: TI (tecnologia da informação), refrigeração e energia. Portanto, a disponibilidade dos serviços hospedados na nuvem pode ser afetada por vários fatores, como falhas de componentes de TI, ou falhas em componentes de refrigeração ou de energia; além disso, a falha de um desses subsistemas pode acarretar no mal funcionamento de outro. Entre todos os subsistemas que compõem um *data center*, a infraestrutura de energia é crucial, pois a mesma tem o papel de fornecer energia de maneira ininterrupta para os demais subsistemas.

Além das falhas nos subsistemas físicos, também é comum ocorrer falhas no nível lógico, atingindo diretamente os *softwares* e serviços que estão hospedados nos servidores do *data center*. Essas falhas no nível lógico podem ser mitigadas, por exemplo, pelo uso do mecanismo de *checkpoint*, que é responsável por armazenar informações sobre o estado de uma aplicação específica ou de uma máquina virtual (*virtual machine* - VM) completa. Enquanto que as falhas no nível físico podem ser mitigadas pelo uso de redundância. No entanto, essa estratégia depende da aquisição de mais equipamentos, o que pode ser um ponto negativo para a sustentabilidade e pode onerar o custo de aquisição e manutenção do *data center*.

Tendo em vista esse cenário, este trabalho propõe um conjunto de arquiteturas da infraestrutura de energia de *data centers* com base em análise de sensibilidade, com o intuito de aumentar sua disponibilidade. Estas arquiteturas serão modeladas utilizando modelos estocásticos (*Stochastic Petri Net* (SPN) e *Reliability Block Diagram* (RBD)). Além das arquiteturas de energia, este trabalho também propõe modelos para representar duas aplicações distintas hospedadas na nuvem e o comportamento do mecanismo de *checkpoint* nas aplicações e também foi proposto um modelo genérico para representar as falhas dos equipamentos de TI utilizando RBD. O principal objetivo é analisar como as falhas no subsistema de energia impactam na disponibilidade das aplicações em nuvem, e também verificar como o mecanismo de *checkpoint* pode mitigar tais falhas.

Com base nisso, pode-se pontuar como contribuições do trabalho: as arquiteturas da infraestrutura de energia, que tem justificativas plausíveis para suas modificações baseadas em analise de sensibilidade; os modelos genéricos dos equipamentos de TI; e a utilização de mecanismo de *checkpoint* em aplicações distintas com testes em diferentes tamanhos.

O restante deste artigo está dividido da seguinte forma: os conceitos básicos são apresentados na seção II. Os modelos de disponibilidade serão descritos na seção III, sendo divididos em modelos de infraestrutura de energia, modelos de *checkpoint* e modelo de integração. Os resultados de disponibilidade são apresentados na seção IV. Os trabalhos relacionados são descritos na seção V. E por fim, o artigo é concluído na seção VI.

Élisson Rocha, Universidade de Pernambuco (UPE), Brasil, email: elissonrochaa@gmail.com.

Guto Leoni, Universidade Federal de Pernambuco (UFPE), Brasil, email: guto.leoni@gprt.ufpe.br.

Patricia Takako Endo, Universidade de Pernambuco (UPE), Brasil, email: patricia.endo@upe.br.

II. Conceitos Básicos

A. Infraestrutura de Data Center

Comumente, usa-se o termo *data center* para tratar sobre o espaço onde estão alocados os equipamentos de Tecnologia da Informação (TI), porém sua infraestrutura abrange outros subsistemas além desse. De acordo com [3], a principal função de um *data center* é entregar serviços básicos, como energia e refrigeração para que o provedor possa oferecer serviços computacionais a terceiros.

Dessa forma, os principais subsistemas que compõem um *data center* são: subsistema de energia, subsistema de refrigeração e subsistema de TI. O subsistema de refrigeração é responsável por remover o calor gerado pelos equipamentos de TI com o intuito de mitigar danos. O subsistema de TI é responsável por fornecer recursos computacionais (computação, rede e armazenamento) para as aplicações hospedadas na nuvem [3]. E o subsistema de energia é responsável por fornecer potência para os equipamentos de TI, como para infraestrutura de resfriamento, com frequência e tensão corretas. Essa infraestrutura é o foco da pesquisa e será apresentado mais detalhadamente na próxima subseção.

B. Subsistema de Energia de Data Center

A tarefa desse subsistema é considerada a mais crítica de um *data center* [4]. Segundo [5], existem vários distúrbios que afetam a qualidade da energia, consequentemente, influenciando no bom funcionamento dos equipamentos conectados ao sistema elétrico. A infraestrutura deve alimentar de forma contínua e adequada os equipamentos de TI e equipamentos de resfriamento. Desta forma, são necessários equipamentos auxiliares para essa distribuição ocorrer de forma adequada..

A alimentação elétrica proveniente da concessionária (empresas de fornecimento de energia elétrica) é a energia primária do *data center*. Sabendo que a energia primária pode eventualmente falhar, uma fonte de energia secundária, normalmente um conjunto de geradores, deve estar preparada para manter o *data center* ativo.

Dispositivos *Uninterruptible Power Supply* (UPS) são usados para proteger as cargas elétricas de alterações, tais como pequenas interrupções inesperadas, mudança de frequência ou picos de tensão, melhorando assim a qualidade da fonte de alimentação. Os *power distribution units* (PDUs) são dispositivos responsáveis pela distribuição da energia elétrica. Os *Automatic Transfer Switches* (ATS) ou os *Static Transfer Switches* (STS) fazem a troca de qualquer utilitário de energia que falhou durante o processo de entrega em baterias reservas [6].

O funcionamento desses dispositivos são cruciais para que a energia chegue com qualidade nos equipamentos finais. Qualquer problema nesse subsistema faz com que os demais subsistemas fiquem indisponíveis e, consequentemente, as aplicações que estão executando tornam-se indisponíveis. Para mitigar falhas que acontecem no nível físico, existem mecanismos de alta disponibilidade implementados no nível lógico, como o mecanismo de *checkpoint* descrito a seguir.

C. Mecanismo de Checkpoint

O mecanismo de *checkpoint* é o processo de replicação que armazena informações necessárias do estado atual de uma aplicação para manter o mesmo em execução posteriormente [7]. Com isso, em caso de falha, a aplicação pode ser reinicializada com perda mínima de dados a partir do último *checkpoint* salvo.

Por exemplo, uma aplicação sendo executada em um servidor e que recebe requisições dos usuários é considerada ativa, enquanto pode haver vários servidores em *standby*. A replicação baseada em *checkpoint* fornece atualizações do servidor ativo para os servidores *standby*. Assim, em caso de falha do servidor ativo, uma instância da aplicação pode ser recuperada e ativa de um dos servidores em *standby*.

Nabi, Toeroe e Khendek [8] apresentam três modos de recuperação de um *checkpoint* após uma falha: *cold*, *warm* e *hot*, todos baseados no padrão *Service Availability Forum* (SAF) [9], como mostra a Figura 1. O gerenciador coordena a criação e exclusão de *checkpoints*.

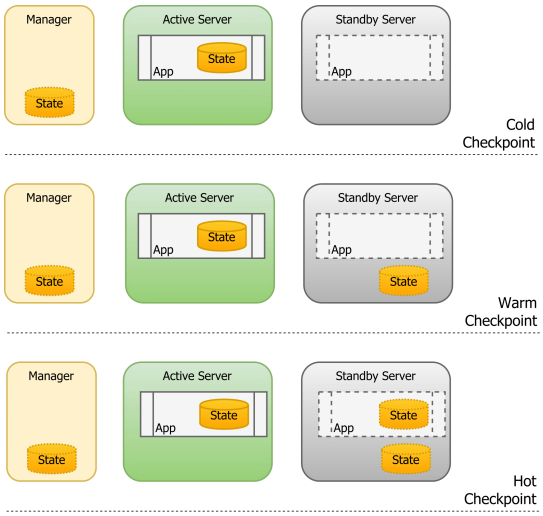


Fig. 1. Modos de recuperação do mecanismo de *checkpoint*.

No modo *cold*, a aplicação ativa (hospedada no *active server*) envia periodicamente seu estado atual para ser armazenado no gerenciador (*manager*), e o servidor *standby* solicita esta informação ao gerenciador, quando houver uma falha e a aplicação precisar ser reinicializada no servidor *standby*.

No modo *warm*, o estado atual da aplicação é armazenado no gerenciador e no servidor *standby*, porém, a aplicação que está hospedada no servidor *standby* só recebe esta informação quando houver uma falha e a mesma precisar ser reinicializada no servidor *standby*.

Similar ao *warm*, a diferença do modo *hot* é que o armazenamento do estado atual da aplicação ativa é feito em todos os componentes: gerenciador, servidor *standby* e aplicação hospedada no servidor *standby*. Portanto, o tempo de recuperação do modo *hot* é mais curto que os demais modos, porém ele necessita de mais recursos computacionais para se tornar operacional.

D. Realibility Block Diagram

Realibility Block Diagram (RBD) é um modelo combinatório que inicialmente foi proposto para analisar a confiabilidade de um sistema por diagrama de blocos. Esse diagrama de blocos representa, de forma gráfica, os componentes que podem determinar o estado geral do sistema, com base no estado de cada componente. A estrutura RBD define através de interações lógicas quais combinações de componentes, com falha ou não, resulta em um sistema ativo. Posteriormente, este formalismo foi estendido para a análise de disponibilidade e manutenção [10].

Os componentes básicos do RBD são: vértice de origem, vértice de destino, blocos (representando os componentes do sistema), e arcos, conectando os blocos e os vértices. Um sistema está disponível quando há um caminho do vértice de origem até o vértice de destino.

Basicamente, existem duas maneiras de organizar os componentes do sistema: em série e em paralelo. Quando os blocos estão organizados em série, é necessário que todos estejam operantes para o sistema, como um todo, possa ser considerado em operação. Quando estão arranjados em paralelo, existe uma redundância de componentes onde precisa-se de, no mínimo, um bloco ativo para manter o conjunto disponível.

Embora RBD possa ser usado para representar a relação de dependência entre os componentes do sistema, esta técnica não permite que os sistemas sejam mais complexos [11]. Para modelar sistemas mais complexos, pode-se usar Redes de Petri.

E. Redes de Petri

Redes de Petri utilizam uma representação matemática para modelar sistemas, possibilitando análises da estrutura e do comportamento dinâmico do modelo [12].

Graficamente, uma Rede de Petri é composta por círculos (círculos brancos representam lugares, e círculos pretos representam *tokens*), retângulos (transições) e arcos (ver Figura 2). Lugares descrevem componentes do sistema e *tokens* são atribuídos em lugares para representar o estado do sistema em um determinado momento. Existem dois tipos de transições: temporizada (Figura 2.b) e imediata (Figura 2.c). A transição temporizada é ativada através de um parâmetro de tempo que segue uma função de distribuição de probabilidade, geralmente exponencial. As transições imediatas são ativadas instantaneamente, após serem habilitadas por funções de guarda. Caso duas ou mais transições imediatas sejam habilitadas ao mesmo tempo, pode ser definida uma prioridade entre elas. Uma Rede de Petri que possua transições temporizadas que sigam processos estocásticos é chamada de *Stochastic Petri Net* (SPN).

F. Análise de Sensibilidade

A análise de sensibilidade é um método que procura identificar quais fatores mais influenciam um determinando comportamento de um sistema [13], [14]. Com a variação dos parâmetros de entrada, verifica-se como se comportam as variáveis de saída, podendo assim, descobrir pontos cruciais do processo.

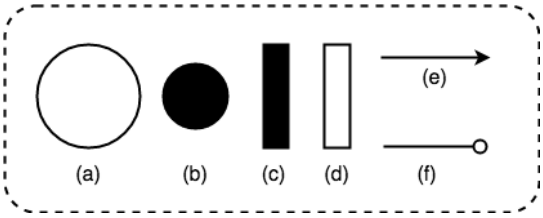


Fig. 2. Componentes Rede de Petri.

Outras vantagens da análise de sensibilidade são o poder de identificar componentes que podem ser retirados do sistema sem grandes perdas nos resultados e orientar possíveis otimizações no sistema ao encontrar gargalos que prejudiquem desempenho ou confiabilidade.

Para obter o índice de sensibilidade de cada componente do sistema a ser analisado, pode-se utilizar diferentes métodos, como *Design of Experiments* ou *Sensitivity Indices*. Neste trabalho, utilizou-se o método chamado de *Sensitivity Indexis*, uma vez que usa-se a técnica de diferença percentual, calculando a variação percentual correspondente na métrica escolhida. Essa técnica necessita de um valor mínimo e máximo de cada componente. Com isso, calcula-se o índice de sensibilidade de acordo com a Eq. 1.

$$S\theta(Y) = \frac{max\{Y(\theta)\} - min\{Y(\theta)\}}{max\{Y(\theta)\}} \tag{1}$$

onde $\max\{Y(\theta)\}$ and $\min\{Y(\theta)\}$ são, respectivamente, os valores de saída mínimo e máximo calculados ao variar o parâmetro de entrada sobre o intervalo dos valores de interesse [15].

III. Modelos de disponibilidade

Esta seção está dividida em três modelos distintos: infraestrutura de energia (subseção III-A), aplicação com mecanismo de *checkpoint* (subseção III-B) e infraestrutura de TI com a integração dos três modelos (subseção III-C).

A. Infraestrutura de Energia

A infraestrutura de energia considerada nesse artigo é composta por vários componentes: *AC Source* (AC), *Generator* (G), *Automatic Transfer Switch* (ATS), *Voltage Panel* (VP), *Uninterruptible Power Supply* (UPS), *Transformer* (T), *Sub-Panel* (SP) e *Junction Box* (JB), e podem ser organizados em série [6]. A Figura 3 apresenta nossa arquitetura básica de energia (chamada A0), sem redundância. Neste caso, a infraestrutura de energia falha quando um componente do sistema falha, com exceção do AC, pois o G é responsável por substituí-lo em caso de falha. Dessa forma, consideramos que a arquitetura A0 é vulnerável.

O modelo SPN correspondente a A0 é mostrado na Figura 4. Cada equipamento é representado por um *building block* com dois lugares e duas transições. Os lugares denominados $componente\_UP$ e $componente\_DOWN$ representam componentes funcionando ou em falha, respectivamente. A transição que representa uma falha tem um valor de *mean*

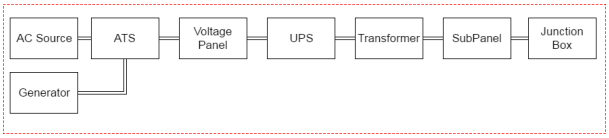


Fig. 3. Arquitetura A0 - Infraestrutura básica de energia sem redundância.

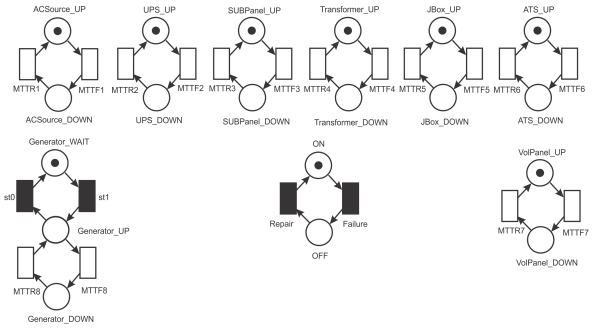


Fig. 4. SPN da Arquitetura A0.

time to failure (MTTF), enquanto a transição que representa um reparo possui valor de *mean time to recovery* (MTTR).

Apenas o componente G apresenta uma diferença: além do *building block* simples mencionado anteriormente, o G apresenta mais um lugar, chamado $WAIT$ e duas transições imediatas ($st0$ e $st1$). O G só funciona quando o *AC Source* falha; logo, se isso não acontecer, ele fica aguardando, no estado $WAIT$. Quando o *AC Source* falhar, o G é ativado (pela transição $st1$), e quando o *AC Source* retorna, o G retorna ao estado de espera (transição $st0$).

As funções de guarda presentes nas transições imediatas são descritas na Tabela I. Além dos componentes da arquitetura de energia, o estado do sistema geral é representado com outro *building block*, com dois lugares, ON e OFF e duas transições imediatas com funções de guarda, $Failure$ E $Repair$.

TABELA I
FUNÇÕES DE GUARDA DE TRANSIÇÃO IMEDIATA DO SPN - A0

Identificação	Função de Guarda
$st0$	#ACSource_DOWN=0
$st1$	#ACSource_DOWN=1

Com o intuito de melhorar essa arquitetura, foi realizada uma análise de sensibilidade utilizando a ferramenta Mercury[1], para entender qual componente tem um maior impacto na disponibilidade geral do sistema de energia.

Os valores de MTTF e MTTR de cada componente estão apresentados na Tabela II. Para se obter os valores máximos e mínimos utilizados na análise de sensibilidade, variou-se os valores padrões para mais e para menos em 30%.

A Tabela III mostra os resultados da análise de sensibilidade da arquitetura A0, com os cinco componentes que têm mais impacto na disponibilidade. O ATS apresenta o maior índice

[1]http://www.modcs.org/?page id = 1397

TABELA II
VALORES MTTF E MTTR DOS COMPONENTES DO SISTEMA DE ENERGIA (POR [11])

Componente	MTTF (em hora)	MTTR (em hora)
AC Source	4380	8
Generator	2190	8
UPS	50000	8
SubPanel	304000	8
Transformer	282581	8
Junction Box	5244000	8
Voltage panel	304000	8
ATS	48076	8

(0.0000383942371), o que significa que ele impacta mais na disponibilidade da infraestrutura de energia.

TABELA III
ANÁLISE DE SENSIBILIDADE DA ARQUITETURA A0

Componente	Índice de sensibilidade
ATS	$3.83942371 \times 10^{-5}$
UPS	$3.69175256 \times 10^{-5}$
Voltage Panel	6.0728889×10^{-6}
SubPanel	6.0728813×10^{-6}
AC Source	7.6745982×10^{-7}

Então, considerando o resultado da análise de sensibilidade, propomos a arquitetura A1 com dois componentes ATS, como mostrado na Figura 5.

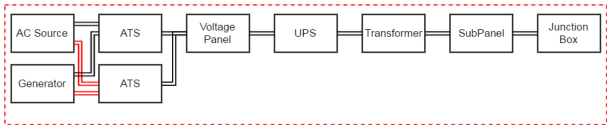


Fig. 5. Arquitetura A1 - Infraestrutura de energia com redundância no ATS.

O modelo SPN correspondente à arquitetura A1 é semelhante ao A0, como mostrado na Figura 6. A diferença é que mais um *token* é adicionado no lugar ATS_UP, representando a redundância ATS no modelo. Com essa alteração, as políticas de concorrência também precisam ser alteradas, no lugar que representa o ATS é modificado de serviço único para *infinite server*, pois entende-se que os dois tokens que representa o ATS podem falhar ao mesmo tempo.

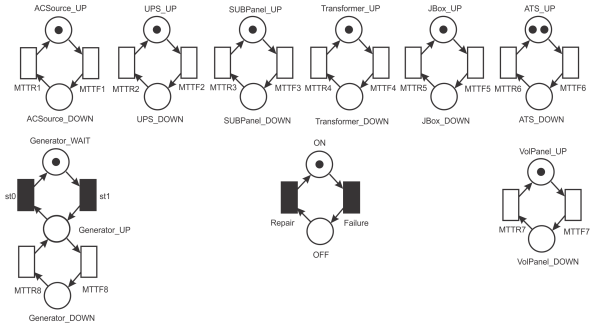


Fig. 6. SPN da Arquitetura A1.

Para propor novas arquiteturas de energia, seguimos uma metodologia baseada na análise de sensibilidade, replicando o componente que atingiu o maior índice, conforme descrito para propor a arquitetura A1. Assim, as arquiteturas A2 e A3 foram propostas como ilustrado nas Figuras 7 e 8, respectivamente.

Os modelos SPN de A2 e A3 são semelhantes a A1, onde para cada componente de redundância, temos dois *tokens* para representá-lo. Na arquitetura A2, temos ATS e UPS com redundância (Figura 7); e na arquitetura A3, temos ATS, UPS, VP, T e SP com redundância (Figura 8). Na arquitetura A3, a redundância é em quase todos os componentes, já que o G serve como redundância para o *AC Source*. Com isso, só o *Junction Box* fica sem redundância, pois, como visto na Tabela II, o seu MTTF é muito alto, fazendo com que o componente tenha um pequeno impacto na disponibilidade.

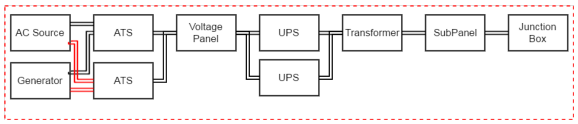


Fig. 7. Arquitetura A2 - infraestrutura de energia com redundância ATS e UPS.

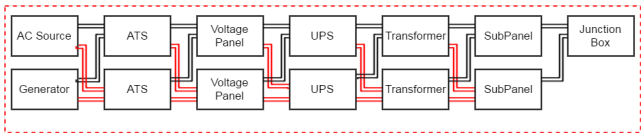


Fig. 8. Arquitetura A3 - Infraestrutura de energia com redundância ATS, VP, UPS, T e SP.

B. Aplicação com Modelo de Checkpoint

A Figura 9 apresenta o modelo SPN que representa o mecanismo de *checkpoint*. Neste modelo, a aplicação que está hospedada no *data center* está representada por dois lugares (APP_UP e APP_DOWN) e uma transição $MTTF10$, onde nela consta o tempo médio de falha do componente. Neste modelo, ao invés de considerar o MTTR da aplicação, utilizou-se os valores de recuperação do mecanismo de *checkpoint*.

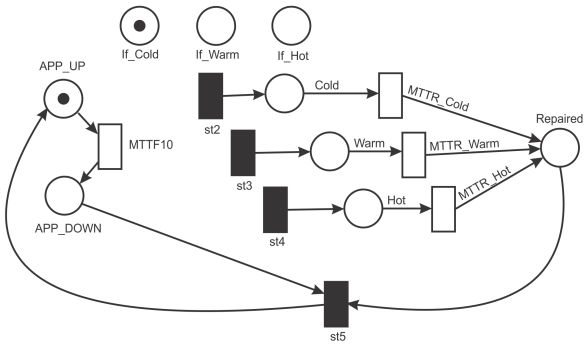


Fig. 9. Modelo de Checkpoint no SPN.

Quando a aplicação falha, inicia-se o mecanismo de *checkpoint*. Primeiro, através dos lugares If_Cold, If_Warm e If_Hot, é possível configurar o modo de recuperação a ser utilizado, isso porque para cada modo de recuperação há um valor de MTTR diferente, $MTTR\_Cold$, $MTTR\_Warm$ e $MTTR\_Hot$, respectivamente. Definimos funções de guarda para garantir esse comportamento, como pode-se ver na Tabela IV.

TABELA IV
FUNÇÃO DE GUARDA DO MODELO DE CHECKPOINT

Indentificação	Função de Guarda
st2	(#App_UP=0)AND(#If_Cold=1)AND(#Cold=0)AND(#Repaired=0)
st3	(#App_UP=0)AND(#If_Warm=1)AND(#Warm=0)AND(Repaired=0)
st4	(#App_UP=0)AND(#If_Hot=1)AND(#Hot=0)AND(#Repaired=0)

Dependendo do modo de recuperação configurado, o *token* vai para o lugar correspondente ($Cold$, $Warm$ ou Hot) e habilita a transição MTTR correspondente ($MTTR\_Cold$, $MTTR\_Warm$ ou $MTTR\_Hot$), a aplicação é reparada e retorna ao estado APP_UP.

C. Integração com o Modelo da Infraestrutura de Energia

O mecanismo de *checkpoint* anteriomente apresentado se refere à aplicação, porém a aplicação está hospedada em uma infraestrutura de TI. A infraestrutura de TI será modelada através de RBD, como pode ser visto na Figura 10.

Fig. 10. Infraestrutura de TI no RBD.

A infraestrutura de TI será formada por *hardware* (HW), sistema operacional (OS) e máquina virtual (VM). Os valores de MTTF e MTTR dos componentes são citados na Tabela V.

TABELA V
MTTF E MTTR DOS COMPONENTES DE TI. POR [16], [17]

Componente	MTTF (em hora)	MTTR (em hora)
Hardware	8760	1.667
Sistema Operacional	1440	1
Máquina virtual	1880	0.167
Aplicação de video por demanda	217.77	0.92633
Aplicação de Livraria Digital	6865.3	0.167

Para a integração dos modelos, foram gerados os valores de MTTF e MTTR dessa infraestrutura de TI, 75,982 e 0,7264, respectivamente, e adicionado um componente simples no SPN, como poderá ser visto na Figura 11.

A Figura 11 apresenta o modelo integrado, contendo o modelo da infraestrutura de energia A0, a infraestrutura de TI, a aplicação e o mecanismo de *checkpoint*.

Na integração é preciso atualizar as funções de guarda das transições imediatas $Failure$ e $Repair$ (Tabela VI). Para calcular a disponibilidade da aplicação, aplica-se a fórmula: $P\#ON > 0$.

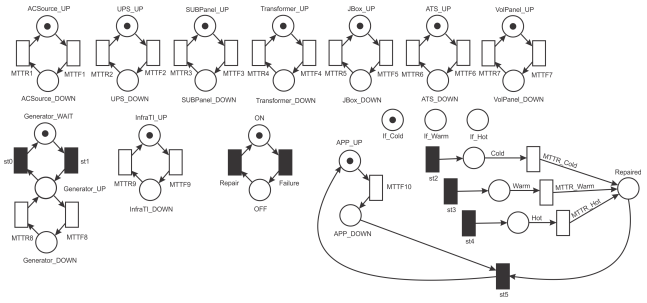


Fig. 11. SPN da integração.

TABELA VI
FUNÇÕES DE GUARDA DO SISTEMA GERAL

Identificação	Função de guarda
$Failure$	*((#ACSource_UP=0)AND(#Generator_UP=0))OR (#ATS_UP=0)OR(#VoltagePanel_UP=0)OR (#UPS_UP=0)OR (#Transformer_UP=0)OR (#SubPanel_UP=0)OR (#JunctionBox_UP=0)OR(#InfraTI_UP=0)OR(#APP_UP=0)*
$Repair$	*((#ACSource_UP>0)OR(#Generator_UP>0))AND (#ATS_UP>0)AND(#VoltagePanel_UP>0) AND(#UPS_UP>0) AND(#Transformer_UP>0)AND(#SubPanel_UP>0) AND(#JunctionBox_UP>0)AND(#InfraTI_UP>0)AND(#APP_UP>0)*

IV. RESULTADOS

Dois experimentos foram considerados: *(i)* resultados de comparação das infraestruturas de energia com aplicações distintas e *(ii)* resultados considerando o mecanismo de *checkpoint* para recuperação das aplicações em falha. A Tabela VII apresenta os parâmetros utilizados nas análises dos experimentos.

TABELA VII
PARÂMETROS E NÍVEIS

	Parâmetros	Níveis
Experimento i	Arquitetura de energia	A0, A1, A2 e A3
	Aplicação	VoD e Livraria Digital
Experimento ii	Arquitetura de energia	A0, A1, A2 e A3
	Mecanismo de *checkpoint*	Cold, warm e hot
	Tamanho do *checkpoint*	1MB e 100MB
	Aplicação	VoD e Livraria digital

Para entender como as falhas da infraestrutura de energia e do mecanismo de *checkpoint* impactam em diferentes tipos de aplicativos, foram consideradas duas aplicações: *(i)* Video por Demanda (VoD) e *(ii)* livraria digital.

Para calcular a disponibilidade do experimento 1, utilizou-se o modelo SPN descrito na Seção III-A. Os valores de MTTF e MTTR dos componentes do subsistema de energia foram mostrados anteriormente na Tabela II e os MTTF e MTTR dos componentes da infraestrutura de TI e das aplicações na Tabela V.

Para cada aplicação, foram realizadas análises estacionárias considerando cada infraestrutura de energia (de A0 a A3). Uma vez calculada a disponibilidade, A, pode-se determinar o valor de $downtime$ usando a Eq. 2.

$$downtime = (1 - A) * 8760 \quad (2)$$

A Tabela VIII exibe os resultados de disponibilidade e *downtime* da aplicação VoD e da aplicação de Livraria Digital, para cada tipo de arquitetura de energia.

TABELA VIII
RESULTADOS DE DISPONIBILIDADE DA ANÁLISE ESTACIONÁRIA

Aplicação	Arquitetura	disponibilidade (%)	Downtime (h/ano)
VoD	A0	0,9942483544998598	49,86676649
	A1	0,994413770365131	48,43261093
	A2	0,994572852067457	47,05337258
	A3	0,994790679085393	45,16481233
Livraria Digital	A0	0,9984533116840936	13,4097877
	A1	0,998619429611376	11,96954527
	A2	0,998779181193613	10,58449905
	A3	0,998997928688343	8,687958272

Com pode ser verificado na Tabela VIII, considerando a aplicação VoD, é possível observar uma pequena queda no tempo de inatividade de 49,86 h/ano para 45,16 h/ano. Na livraria digital houve uma melhora de 13,40 h/ano para 8,68 h/ano.

A partir da análise estacionária, observou-se que o *downtime* da aplicação VoD foi otimizado cerca de 9,5% quando mudou-se da arquitetura A0 para A3; e o *downtime* da livraria digital melhorou cerca de 35%.

Para o experimento 2, os valores de MTTR do mecanismo de *checkpoint* para os tamanhos dos *checkpoints* estão aprsentados na Tabela IX. Os valores foram retirados de [18] e representam valores reais medidos a partir de experimentos em protótipo.

TABELA IX
VALORES DE MTTR DO MECANISMO DE *checkpoint* (EM SEGUNDOS) [18]

Mecanismo / Tamanho	1MB	100MB
Cold	0,3955	4,321
Warm	0,383	3,8621
Hot	0,2515	0,2677

Os resultados de disponibilidade e *downtime* serão apresentados por tamanho do *checkpoint* (1MB e 100MB). Sendo assim, as Figuras 12 e 13 apresentam disponibilidade e *downtime* por mecanismo de *checkpoint* e por arquitetura de energia, considerando que o tamanho do *checkpoint* é 1MB. Na figura 12(a) são apresentados os resultados da disponibilidade da aplicação VoD, na Figura 12(b) a disponibilidade da aplicação de livraria digital. Na Figura 13(a) o *downtime* da aplicação de VoD, e por fim, na Figura 13(b) o *downtime* da aplicação de livraria digital.

Neste caso, pode-se perceber que os valores são muito próximos, independente do mecanismo de *checkpoint*. O que mais impacta nos resultados é a arquitetura de energia do *data center*. Como esperado, a arquitetura com maior nível de redundância (A3) obtêm os melhores resultados.

As Figuras 14 e 15 apresentam os resultados de disponibilidade e *downtime* quando o tamanho do *checkpoint* é 100MB.

Como pode-se notar, novamente a arquitetura A3 obteve o melhor desempenho, e independente do mecanismo de *checkpoint* e do tamanho do *checkpoint* os valores de disponibilidade são poucos alterados.

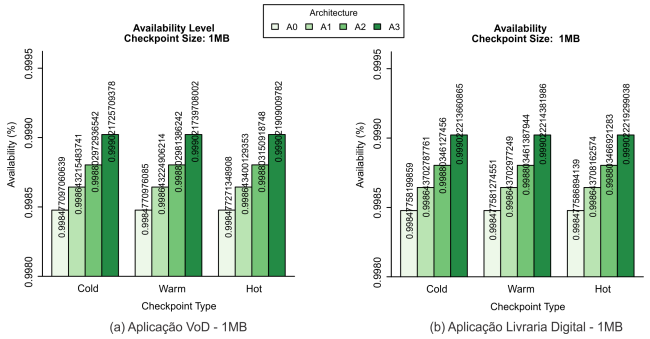


Fig. 12. Disponibilidade com *checkpoint* de 1MB.

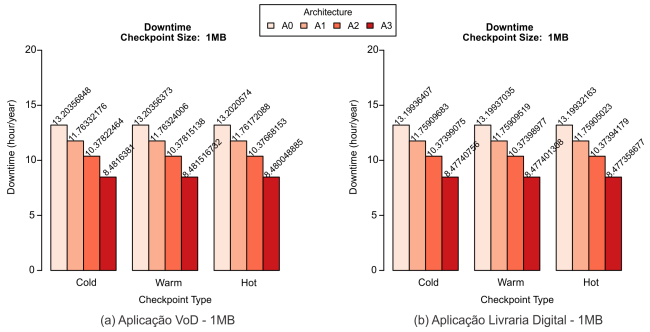


Fig. 13. *Downtime* (horas/ano) com *checkpoint* de 1MB.

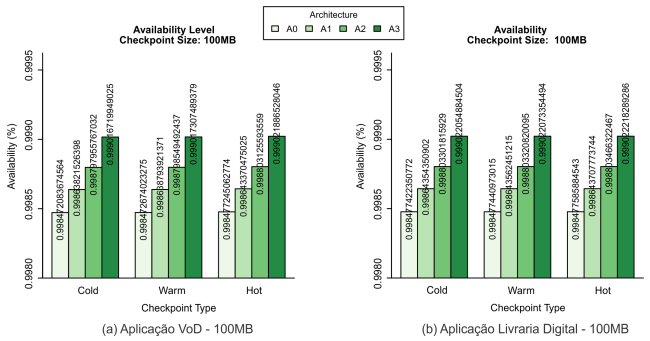


Fig. 14. Disponibilidade com *checkpoint* de 100MB.

Na aplicação de VoD, o maior tempo de inatividade está na arquitetura A0 com mecanismo de checkpoint *cold* e com *checkpoint* de tamanho 100MB, em torno de 13 horas e 15 minutos por ano. Já o menor *downtime* é em torno de 8 horas e 29 minutos por ano e foi encontrado na arquitetura A3 com mecanismo de *checkpoint hot* e com *checkpoint* de tamanho 1MB.

Para a aplicação de livraria digital, os valores foram muito próximos ao de VoD: tem-se o maior *downtime* de 13 horas e 12 minutos por ano e o menor de 8 horas e 28 minutos por ano, aproximadamente. Esses valores são encontrados nas arquiteturas A0 e A3 com mecanismo de *checkpoint cold* e *hot* com *checkpoint* de 100MB e 1MB, respectivamente.

A. Discussão

A partir dos resultados da análise estacionária, a discussão será realizada a partir de duas perspectivas: *(a)* o impacto da arquitetura de energia, e *(b)* o impacto dos mecanismos de *checkpoint*.

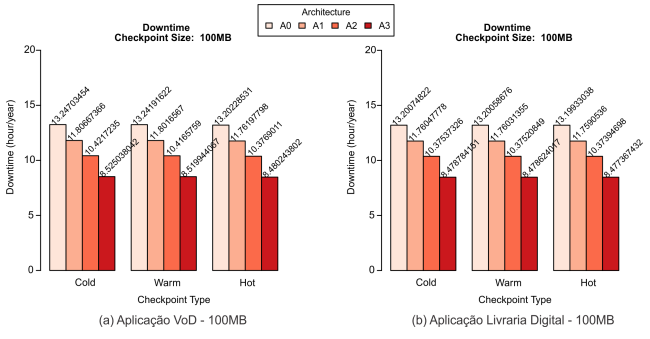


Fig. 15. *Downtime* (horas/ano) com *checkpoint* de 100MB.

Como foi observado, a arquitetura de energia tem um grande impacto na disponibilidade do serviço. À medida que aumenta-se o nível de redundância da arquitetura, consegue-se melhores resultados. Se considerar um dos cenários onde a arquitetura A0 e A3 usam o mesmo mecanismo de *checkpoint*, e independente do tamanho do *checkpoint*, a arquitetura A3 apresenta uma redução de aproximadamente 4,74 horas no *downtime* anual com relação à arquitetura A0.

Considerando os mecanismos de *checkpoint*, observa-se que o uso desse mecanismo equivale a um alto ganho quando a aplicação possui um MTTF baixo. Isto ocorre porque mesmo com a aplicação falhando muito durante um ano, o tempo de recuperação da aplicação é bastante pequeno (milésimos, e em alguns casos segundos). Um exemplo disso é a aplicação VoD: sem o mecanismo de *checkpoint*, a mesma apresenta um *downtime* de 49,867 horas/ano para arquitetura A0 e 45,165 horas/ano para arquitetura A3, e com o mecanismo de *checkpoint* esses valores caem para 13,204 horas/ano e 8,482 horas/ano (dependendo do tamanho do *checkpoint*), respectivamente.

Quando a aplicação possui um MTTF alto, o mecanismo de *checkpoint* melhora seus resultados, porém de forma suave. Um exemplo disso é a aplicação de livraria digital, que sem o mecanismo de *checkpoint* obtém 13,410 horas/ano de tempo de inatividade na arquitetura A0 e 8,688 horas/ano para A3. Quando adotado o mecanismo de *checkpoint* esses valores caem discretamente para 13,202 horas/ano e 8,48 horas/ano, respectivamente.

V. Trabalhos Relacionados

Alguns trabalhos na literatura já apresentaram avaliações de disponibilidade de *data center*, algumas voltadas para componentes de TI, outras focados na infraestrutura de resfriamento e outras em infraestrutura de energia. Por exemplo, os Andrade et al. [15] e Silva et al. [19] usaram o SPN para modelar a recuperação de desastres como um serviço (DRaaS) e também forneceu uma análise de sensibilidade.

Andrade et al. [15] observaram que a adoção de uma solução de recuperação de desastres reduz significativamente o número de requisições perdidas, chegando a 79% de melhoria em alguns casos. Com isso, o serviço consegue atender um número

maior de requisições, evitando problemas para a empresa que o fornece devido à quebra de contratos.

Callou et al. [11] propuseram um conjunto de modelos para quantificar o impacto, custo e disponibilidade de energia das infraestruturas de energia e refrigeração dos centros de dados. Para isso, os autores usaram SPN e RBD para avaliar a disponibilidade e um modelo de fluxo para estimar o custo de energia e os custos das arquiteturas. Eles concluíram que a utilização de redundância é uma estratégia vantajosa, como esperado, mas tem um impacto negativo sobre a sustentabilidade ambiental e seus custos.

No trabalho de Gomes [18] é realizada uma avaliação de desempenho de serviços de *checkpoints* para aplicações de vários níveis, medindo o tempo de *checkpoint*, *failover* e consumo de recursos. Uma aplicação *state-aware*, na qual manipula seu estado, foi desenvolvida para permitir o *checkpoint* no nível da aplicação. O *checkpoint* a nível do sistema foi desenvolvido integrando ferramentas existentes. Os resultados mostram que o *checkpoint* no nível do sistema apresenta piores tempos de *failover* e *checkpoint* em relacão à solução do nível da aplicação

Já Marwah et al [12] apresentaram uma abordagem para verificar o impacto sustentável das arquiteturas do *data center*. A disponibilidade é medida utilizando modelos SPN, enquanto uma abordagem de avaliação do ciclo de vida é usada para quantificar o impacto na sustentabilidade. Foram propostas cinco arquiteturas e observou-se que o custo total obteve um aumento de quase 80% com a introdução de um UPS.

A principal contribuição deste trabalho é a análise de diferentes arquiteturas de energia de um *data center* que hospeda dois tipos diferentes de aplicativos: VoD e livraria digital, além de possuir mecanismo de *checkpoint* nas aplicações de teste.

VI. Conclusão e trabalhos futuros

Este trabalho analisou o impacto da infraestrutura de energia na disponibilidade de aplicações hospedadas na nuvem. Foram propostas quatro arquiteturas de infraestrutura de energia, da mais simples e sem redundância, até a mais complexa, com maior número de componentes replicados. Vale ressaltar que as evoluções feitas de uma arquitetura para outra foram baseadas em análise de sensibilidade.

Analisou-se o impacto das infraestruturas de energia na disponibilidade de aplicações em vários cenários. Os cenários variaram entre quatro arquiteturas de energia, dois tipos de aplicação, três modos de *checkpoint* e dois tamanhos de *checkpoint*. Observou-se que o uso do mecanimo de *checkpoint* equivale a um ganho alto na disponibilidade, porém a diferença entre os modos de mecanismos de *checkpoint* é muito pequena.

Como trabalhos futuros, planeja-se detalhar os componentes de um *data center* de nuvem, considerando os demais subsistemas, bem como, analisar o custo das arquiteturas e medir o retorno sobre o investimento (ROI) [20] para cada um deles.

Referências

[1] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[2] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," *IEEE network*, vol. 25, no. 4, 2011.

[3] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[4] A. B. G. Frigo, "Infraestrutura de data center e suas tendências com foco em eficiência energética," 2015.

[5] J. Crepaldi, M. M. Amoroso, and O. H. A. Junior, "Analysis of the topologies of power filters applied in distributed generation units-review," *IEEE Latin America Transactions*, vol. 16, no. 7, pp. 1892–1897, 2018.

[6] G. R. d. A. Callou, "Assessment to support the planning of sustainable data centers with high availability," 2013.

[7] D. Singh, J. Singh, and A. Chhabra, "High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms," in *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*. IEEE, 2012, pp. 698–703.

[8] M. Nabi, M. Toeroe, and F. Khendek, "Availability in the cloud: State of the art," *Journal of Network and Computer Applications*, vol. 60, pp. 54–67, 2016.

[9] OPENSAF. (2014) Opensaf overview. [Online]. Available: ¡http://sourceforge.net/projects/opensaf/files/docs/opensaf-documentation-4.4.1.tar. gz/download¿

[10] P. Maciel, K. Trivedi, and D. Kim, "Dependability modeling in: Performance and dependability in service computing: Concepts, techniques and research directions," *Hershey: IGI Global, Pennsylvania, USA*, vol. 13, 2010.

[11] G. Callou, P. Maciel, D. Tutsch, J. Ferreira, J. Araújo, and R. Souza, "Estimating sustainability impact of high dependable data centers: a comparative study between brazilian and us energy mixes," *Computing*, vol. 95, no. 12, pp. 1137–1170, 2013.

[12] M. Marwah, P. Maciel, A. Shah, R. Sharma, T. Christian, V. Almeida, C. Araújo, E. Souza, G. Callou, B. Silva *et al.*, "Quantifying the sustainability impact of data center availability," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 64–68, 2010.

[13] P. M. Frank, *Introduction to system sensitivity theory*. Academic press New York, 1978, vol. 1.

[14] D. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models," *Environmental monitoring and assessment*, vol. 32, no. 2, pp. 135–154, 1994.

[15] E. Andrade, B. Nogueira, R. Matos, G. Callou, and P. Maciel, "Availability modeling and analysis of a disaster-recovery-as-a-service solution," *Computing*, pp. 1–26, 2017.

[16] R. Melo, M. C. Bezerra, J. Dantas, R. Matos, I. Melo, and P. Maciel, "Video on demand hosted in private cloud: Availability modeling and sensitivity analysis," in *Dependable Systems and Networks Workshops (DSN-W), 2015 IEEE International Conference on*. IEEE, 2015, pp. 12–18.

[17] J. Araujo, P. Maciel, M. Torquato, G. Callou, and E. Andrade, "Availability evaluation of digital library cloud services," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*. IEEE, 2014, pp. 666–671.

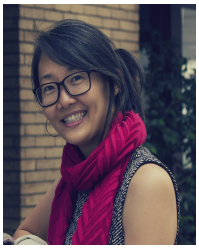
[18] D. Gomes, "Performance evaluation of checkpoint services for multi-tier stateful applications," 2016.

[19] B. Silva, P. Maciel, E. Tavares, and A. Zimmermann, "Dependability models for designing disaster tolerant cloud computing systems," in *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*. IEEE, 2013, pp. 1–6.

[20] R. Caricimi *et al.*, "Economic analysis for small hydroelectric power plant using extended multi-index methodology-an approach stochastic by the monte carlo simulation," *IEEE Latin America Transactions*, vol. 16, no. 8, pp. 2184–2191, 2018.

Élisson da Silva Rocha é formado em Bacharelado em sistemas de Informação pela Universidade de Pernambuco (UPE). Suas áreas de interesse são: cloud computing e modelagem.

Guto Leoni atualmente é aluno de Doutorado em Ciência da Computação na Universidade Federal de Pernambuco (UFPE), possui graduação em Sistemas de Informação pela Universidade de Pernambuco (UPE), e mestrado pela Universidade Federal de Pernambuco. Suas áreas de interesse são: cloud computing, fog computing e modelagem.



Patricia Takako Endo é professora adjunta da Universidade de Pernambuco (UPE) e atualmente é pesquisadora de pós-doutorado na Dublin City University (DCU). Possui doutorado em Ciência da Computação pela Universidade Federal de Pernambuco (UFPE) e suas áreas de interesse são: cloud computing, gerenciamento de recursos, fog computing e modelagem.