2
Descrição do Problema

A ferramenta BLAST é a mais utilizada quando se trata de comparar biosseqüências, seja de nucleotídeos ou proteínas(25). BLAST usa técnicas com algoritmos de programação dinâmica e aplicação de heurísticas na comparação entre as seqüências.

Em casos gerais onde pequenos números de seqüências são submetidos como consultas de entrada para o BLAST, o tempo médio de processamento é considerado baixo. O desempenho do BLAST está fortemente vinculado às seqüências de consulta, base de dados e os parâmetros de entrada. Por exemplo, veja a Tabela 2.1 que apresenta execuções do BLAST com três diferentes seqüências de consulta e duas distintas base de dados. Utilizando-se uma base de dados de tamanho inferior ao tamanho da memória primária disponível e submetendo-se uma seqüência de consulta com 100 caracteres, obtém-se o resultado em aproximadamente 5 segundos (execução 1). Com os mesmos parâmetros e base de dados, mas alterando somente a composição no conteúdo da seqüência de consulta, o resultado passa a ser obtido em aproximadamente 13 segundos (execução 2). Aumentando-se a base de dados para quatro vezes maior que a memória primária, e utilizando-se uma seqüência de consulta de 10.000 caracteres, o tempo final chega a 8 minutos (execução 3). Além das execuções apresentadas na Tabela 2.1, verificamos que com algumas alterações nos valores padronizados para os parâmetros do BLAST, as execuções podem durar até duas horas. Essa sensibilidade aos múltiplos fatores mostra que, por mais rápido que ferramenta BLAST seja, o tempo necessário de processamento vai depender das necessidades e da forma de execução.

Em laboratórios de seqüenciamento é normal que milhares de seqüências sejam processadas pelo BLAST, o que pode em muitos casos exigir um longo tempo de processamento se nenhuma estratégia específica for aplicada. Dessa forma, considerando-se o caso geral em que a submissão de duas seqüências de consulta ao BLAST não possuam nenhuma relação entre si, e que em nenhum momento é necessário ter troca de mensagens para que seja gerado o relatório

	Tamanho da Seqüência de consulta	Base de Dados	Tempo (segundos)
execução 1	100	menor que memória primária	5
execução 2	100	menor que memória primária	13
execução 3	10.000	maior que a memória primária	480

Tabela 2.1: *Distintas execuções do BLAST utilizando diferentes seqüências de consulta para cada execução e duas opções de banco de dados.*

final, pode-se considerar essa aplicação como vergonhosamente paralela (25). Pode-se submeter cada seqüência de consulta a computadores diferentes e, para um mesmo banco de seqüências, o resultado será o mesmo que utilizando um único computador, porém com muito mais eficiência.

No final da década de 90, vários trabalhos passaram a utilizar a ferramenta BLAST em agrupamento de computadores de forma paralela ou distribuída, como opção para reduzir o longo tempo de processamento. Inicialmente a estratégia adotada foi a utilização do banco de seqüências replicado em várias máquinas (16, 35, 10).

Uma boa parte das aplicações associadas aos agrupamentos de computadores de alguma forma lida com a necessidade em balancear a carga de trabalho. Isto não foi diferente com o BLAST. Mesmo que o banco de seqüências esteja replicado, a forma de envio das seqüências de consulta deve ser tratada no intuito em que as várias máquinas de trabalho tenham aproximadamente o mesmo tempo de execução. Várias propostas neste sentido foram feitas, desde o pré-processamento das seqüências de consulta até a análise das máquinas de trabalho durante o processamento (28, 37, 11).

Mesmo diante da redução no tempo de execução obtida com bases replicadas, algumas execuções ainda podem exigir longos tempos de processamento. Como esperado, um dos fatores que aumentam significativamente o custo é o processamento em grandes bancos de dados. Com o crescimento desses, no passar dos últimos anos, alternativas têm sido adotadas evitando que seqüências de consultas sejam comparadas inteiramente com bases de dados maiores do que a memória primária disponível.

Assim, surgiram estratégias de execução paralela com a base de dados

fragmentada, isto é, uma parte distinta de toda a base é alocada em diferentes máquinas, permitindo paralelização do acesso ao disco (28, 21, 26).

No processo de avaliação da ferramenta BLAST em ambientes paralelos, várias são as dificuldades de uso e há muitos requisitos importantes que devem ser estudados. Assim sendo, o objetivo deste Capítulo é discutir alguns destes assuntos, descritos nas próximas subseções, a saber:

- desbalanceamento de carga;
- tempo de execução em bancos de dados maiores que a memória primária;
- dificuldades no uso de estratégias fragmentadas;
- atualização do banco de dados durante o processamento; e
- manutenção do sistema.

2.1 Desbalanceamento de Carga

Considerando uma execução do BLAST com várias máquinas trabalhando juntas, o ganho será desperdiçado se a primeira máquina que conclui o processamento o faz em tempo consideravelmente rápido e a que terminar por último demorar muito mais tempo para finalizar. O tempo total de execução paralela será sempre o tempo a contar do início da execução até o tempo da finalização do processo mais lento (15). Em geral, a máquina mais lenta é a última a enviar a seqüência restante para conclusão da execução total. Para que o sistema tenha melhor desempenho e evite manter máquinas ociosas é importante que o tempo de finalização da última máquina esteja o mais próximo possível do tempo final de execução da máquina mais rápida.

Esse controle de tempo é conhecido como balanceamento de carga. O processo de balanceamento deve ocorrer de forma a evitar que o tempo total não seja superior ao caso em que não exista balanceamento algum. Desta forma, com um processamento paralelo temos por objetivo executar aplicações de forma equilibrada, sem que se pague um preço adicional (*overhead*) por isso.

Nesta dissertação, nos referimos ao termo *carga* como sendo o número total de seqüências que a máquina gerente envia à máquina trabalhadora durante todo o processamento. A carga aumenta sempre que uma nova tarefa é submetida à estação de trabalho. Uma tarefa é o conjunto de uma ou mais seqüências submetidas de uma única vez ao BLAST.

Para exemplificar, vamos considerar duas máquinas distintas A e B, que devem receber 20 tarefas de outra máquina dita gerente. Essa última envia diferentes cargas de processamento, sendo 5 tarefas para o máquina A e 15 tarefas para o máquina B e, ao final, ambas terminam ao mesmo tempo. Essa diferença no número de tarefas pode ter ocorrido, pois a máquina gerente teria percebido que a máquina B tem maior poder de processamento. Ao final, as duas máquinas trabalhadoras terminam juntas, mas processam quantidades de carga diferentes.

2.1.1 Balanceamento na ferramenta BLAST

Como discutimos no início deste Capítulo, o resultado das submissões de duas seqüências distintas ao mesmo banco de dados podem ter tempos diferentes (como mostra a Tabela 2.1). Imaginando a execução em lote de milhares de seqüências, o que é bastante comum na prática, a carga em cada máquina de trabalho pode ter enorme variação se não for adotado um processo de balanceamento (35).

Existem vários métodos de balanceamento de carga na literatura (15), classificados principalmente como estáticos e dinâmicos. O primeiro utiliza-se de informações preliminares conhecidas. As informações para controle do balanceamento são obtidas antes mesmo do início da execução e o método define o balanceamento antes do processamento se iniciar. No segundo as informações são obtidas durante o processamento, após o inicio da execução. É possível encontrar estratégias em que os dois tipos são utilizados juntos.

Como em qualquer execução paralela, há alguns fatores externos que agravam o desbalanceamento e configuram um ambiente heterogêneo, como a existência de processos concorrentes nas máquinas e diferentes configurações das mesmas. Em específico ao processamento BLAST há dois outros fatores importantes que causam desvio de balanceamento (também identificado em (12, 6)): o primeiro diz respeito ao tamanho da seqüência de consulta, pois quanto maior, assim será o número de palavras produzidas para futura extensão pelo BLAST. O segundo fator de desbalanceamento é o grau de similaridade entre a seqüência de consulta e as seqüências das bases de dados. Pois na fase em que cada palavra é estendida, quanto mais similar, maior será o tempo de extensão dessa palavra. Em (19) se encontram mais detalhes do funcionamento do BLAST.

O primeiro fator pode ser tratado se for acrescentado um método para ordenar as seqüências de consulta e alocá-las da melhor forma às máquinas trabalhadoras caracterizando um método de balanceamento estático (9). Contudo, o segundo fator de desbalanceamento dificilmente pode ser identificado antes da submissão da consulta. No Capítulo 3 será visto que em alguns casos, o nível de similaridade entre as seqüências é um fator de maior impacto no nível de balanceamento do que a diferença no tamanho. Seqüências mais similares e menores podem ter maior tempo de processamento do que seqüências maiores e pouco similares.

Estes pontos de desbalanceamento intrínsecos ao processamento BLAST estão discutidos em detalhes no Capítulo 3, onde vários testes são realizados para motivar e justificar algumas das estratégias de avaliação paralela com balanceamento de carga aqui propostas.

Granularidade

Quando se planeja obter balanceamento, um fator de análise é o nível de granularidade que cada aplicação possibilita. Isto é, vamos considerar que tenhamos três arquivos com 2 Gb cada um, e devemos submeter cada arquivo para que seja processado ou lido por três distintas máquinas. Considerando que cada arquivo seja enviado para uma máquina e dada a heterogeneidade entre elas, algumas poderão terminar o processamento rapidamente enquanto outras levarão mais tempo. Contudo, se cada arquivo for dividido em 4 partes de 500 MB cada um, teremos um total de 12 arquivos. Desta forma, é possível enviar um arquivo menor para ser processado em cada máquina, e de acordo com o término de cada processamento local, outro arquivo é submetido à máquina que já concluiu sua tarefa. Esta é exatamente a forma com que a ferramenta mpiBLAST(11) efetua balanceamento de carga. Apesar de ser um exemplo simples, pode-se dizer que há diferença de granularidade de carga, em geral pode ser fina (12 arquivos de 500 MB) ou grossa (3 arquivos de 1GB).

Neste caso, a granularidade é o tamanho da unidade de computação (carga) executada (transferida) entre a estação de gerência e as estações de trabalho (15). Uma granularidade grossa pode diminuir o número de mensagens enviadas, reduzindo-se o tráfego na rede, contudo favorece a ocorrência do desbalanceamento de carga. Já a granularidade fina aumenta o número de mensagens enviadas entre as estações, embora facilite o controle do balancea-

mento de carga.

No caso da ferramenta BLAST, para que uma estação de trabalho possa executar uma comparação, será necessário que receba tanto as seqüências de consulta como as da base de dados. Logo, a primeira forma de granularidade possível se refere ao envio das seqüências de consulta. Neste caso, uma máquina de trabalho pode receber várias seqüências e retornar na medida em que finaliza, ou receber todas as seqüências e enviar os resultado uma única vez.

Outra opção de granularidade possível é permitir que partes da base de dados sejam enviadas às estações de trabalho. Desta forma, cada processamento BLAST será executado em segmentos distintos da base de dados.

Ainda, é possível que ambas as formas de granularidade possam ser utilizadas em uma única estratégia mista, que permita variar tanto a carga de seqüências de consulta como os segmentos da base de dados.

A granularidade utilizada para aplicações da ferramenta BLAST está associada às estratégias de balanceamento de carga. Como será visto no Capítulo 4, a maioria dos trabalhos utilizam variações na granularidade nas seqüências de consulta ou nos fragmentos do banco, e não de forma mista.

2.1.2 Tempo de execução em grandes bancos de dados

Um dos principais custos de execução da ferramenta BLAST é o tempo de acesso ao disco (20, 11, 25). O Capítulo 3 mostra alguns testes com a base de seqüências menor e maior que a memória primária. Associado a isso, podemos ver na Figura 2.1 o gráfico da taxa de crescimento nos últimos anos em bancos de dados públicos, como o Genbank (3). Com o aumento de projetos genomas, é possível verificar um crescimento exponencial nos bancos de seqüências.

Alternativas para este tipo problema, além dos ambientes computacionais paralelos, podem variar, como: utilização de *drivers* para gerenciar o acesso ao disco e permitir um escalonamento dos processos BLAST (24) ou ainda compactar os dados das bases visando menor custo de E/S (31).

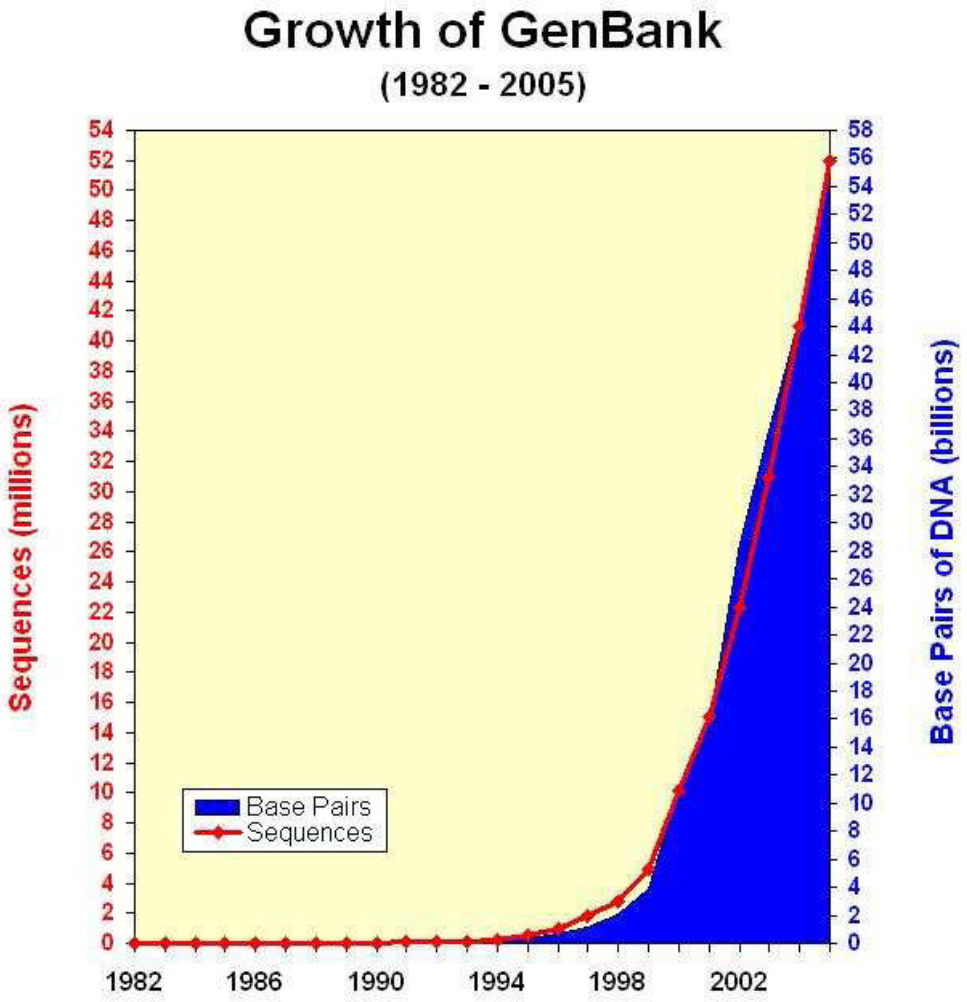

Figura 2.1: *Tendência de crescimento dos banco de dados de seqüências (http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html)*

2.2 BLAST com bases fragmentadas

Com o alto custo de executar a ferramenta BLAST em bases de dados maiores que a memória principal, uma solução adotada em ambientes paralelos é fragmentar toda a base de dados, na tentativa de ajustar o tamanho destes fragmentos à memória principal e, assim, evitar excessivo acesso ao disco. Contudo, esta alternativa possui algumas implicações que abordaremos a seguir.

Em uma estratégia fragmentada, após um ou vários fragmentos serem alocados para uma máquina de trabalho, cada seqüência de consulta é enviada para todas as máquinas que contém fragmentos. Isto é necessário para que

cada seqüência seja comparada a todos os fragmentos, garantindo a correção do processo como um todo. Em cada uma destas comparações será gerado um sub-relatório. Para que o relatório final de cada seqüência seja obtido é necessário que os sub-relatórios gerados na comparação da seqüência com cada fragmento sejam montados. Como cada fragmento possui seqüências distintas, a operação de montagem é quase semelhante a uma operação de fusão (*merge*)(8). Esta operação irá mesclar os sub-relatórios para que ao final se gere um resultado mostrando de forma ordenada as seqüências do banco de dados com maior grau de similaridade em relação à seqüência de consulta.

Comparado à estratégia replicada, o envio de uma seqüência para todas as máquinas e o recebimento dos sub-relatórios, a estratégia rragmentada exige um alto custo de comunicação de dados. Se compararmos o processo de execução entre a estratégia replicada e a estratégia fragmentada, percebemos que esta última possui três passos adicionais (*overhead*) em relação à primeira. Este três passos (exclusivos para as estratégias fragmentadas) serão descritos nas próximas subseções a fim de que cada um possa ser entendido corretamente.

O que se espera da estratégia fragmentada é que ela consiga diminuir o tempo de execução mesmo considerando-se todos estes custos adicionais.

Primeiro "'Custo Adicional"': Pré-processamento do banco de dados

Antes que um banco de dados seja utilizado pela ferramenta BLAST na maioria das estratégias fragmentadas, ele deverá ser fragmentado em pedaços que possam ser alocados às diferentes máquinas. Enquanto a base de dados não sofrer alteração esta tarefa não necessita ser executada novamente. A relevância deste custo esta relacionada com a freqüência em que o usuário irá atualizar a base de dados. Por exemplo, em alguns casos esta alteração pode ocorrer a cada três meses, tornando este custo quase irrelevante.

Estes fragmentos gerados podem estar disponíveis entre as máquinas de diferentes formas. Um exemplo, cada fragmento é colocado em uma máquina, como no exemplo anterior. A principal complicação é a possibilidade de uma estação de trabalho parar de funcionar, tornando o fragmento inacessível, conseqüentemente parando todo o processo.

Outro tipo de disposição dos fragmentos é alocá-los em uma máquina que possua disco compartilhado entre todas as estações de trabalho. Contudo,

várias máquinas farão requisições à mesma estação para diferentes fragmentos, o que irá gerar enorme gargalo no processamento.

Além destas dificuldades, com o crescimento dos bancos de dados (Figura 2.1), um pré-processamento adicional à execução do BLAST se tornará cada vez menos trivial.

Segundo "'Custo Adicional"': Quantidade de mensagens enviadas

Uma seqüência deverá obrigatoriamente ser enviada a todas as estações para que seja comparada contra todos os fragmentos, e o sub-relatório gerado deverá ser recebido pela(s) máquina(s) dita(s) gerente do sistema. Isto aumenta significativamente o número de mensagens na rede. Embora este custo possa não ter tanta relevância considerando as altas taxa de transmissão atuais, o trabalho (28) incentiva o cuidado no uso da rede dado o grande crescimento no número de mensagens.

Terceiro "'Custo Adicional"': Geração do Relatório final

De todos os custos adicionais este é o que possui maior impacto, e pode consumir até 10% de todo o tempo de execução paralelo(21).

Ao final do processamento da estratégia fragmentada os sub-relatórios gerados devem ser montados. Cada sub-relatório contém um ranking das seqüências do fragmento mais similares com as seqüências de consulta. O relatório final deverá ter um único ranking de todas as seqüências pertencentes aos diversos fragmentos. É exatamente neste momento que pode haver uma dificuldade na validação do relatório gerado.

A dificuldade ocorre, pois a ferramenta BLAST utiliza heurísticas para o processo de comparação entre as seqüências, e são fornecidos valores probabilísticos que validam os alinhamentos obtidos, o que constitui as estatísticas de similaridade. Estas são adquiridas principalmente pelas seqüências da base de dados e da consulta. Logo, quando uma seqüência é comparada a um fragmento da base de dados os valores estatísticos são diferentes quanto utilizado todo o banco.

Desta forma, é importante que procedimentos sejam feitos a fim de evitar dados não precisos quando emitidos os relatórios oriundos de processamentos paralelos em bases fragmentadas. Para facilitar o entendimento, e melhor discussão do tema, foi produzido um trabalho em conjunto com esta

dissertação (33) abordando as implicações em bases fragmentadas e as respectivas sugestões. Neste trabalho, mostramos diversos testes que evidenciam os diferentes valores estatísticos em bases fragmentadas e propomos uma forma de utilização dos parâmetros do BLAST que produz valores próximos aos obtidos quando comparado a todo banco de dados.

Ao utilizarmos a estratégia fragmentada nos deparamos com um fino ajuste referente ao número de fragmentos. Pois quanto maior o número de fragmentos gerados, maior será o tempo de montagem dos resultados para cada seqüência de consulta. Em muitos casos, a partir do momento em que os fragmentos já estão completamente alocados na memória principal, o aumento na quantidade de fragmentos pode onerar o tempo de execução total.

2.3 Estratégias sobre bases em processo de atualização

Outro problema, no caso tanto para bancos de dados replicados ou fragmentados, é realizar longos processamentos considerando-se a possibilidade de alteração da base de dados durante a execução do BLAST. Por exemplo, digamos que após um mês de processamento (paralelo ou não) da ferramenta BLAST, sem que o mesmo tenha finalizado, o usuário perceba que a base de dados deve ser incrementada. Atualmente, a alternativa do usuário é parar o processamento, atualizar a base de dados, formatá-la para execução da ferramenta BLAST e reiniciar o processamento todo desde o início. Neste caso o mês de processamento foi em vão.

O ideal para este tipo de problema seria o sistema incorporar a atualização da base de dados aproveitando todo o processamento já ocorrido, e proceder com o restante da comparação contra a base de dados atualizada.

Contudo o problema para isso está no ato de proceder com o restante da comparação, pois com a base de dados alterada o valor que define as estatísticas de alinhamento também foi alterado. Como discutido em (33), estas estatísticas são definidas pelo tamanho da base de dados e o tamanho da seqüência de consulta.

2.4 Robustez do sistema

Na execução do BLAST em larga escala, isto é, comparar uma base de dados contra um grande número de seqüências de entrada, deve-se estar ciente que alguns processamentos podem levar semanas ou meses. Neste sentido, ter um processamento seguro e confiável se torna necessário. Isto é importante para evitar que longas execuções tenham de ser refeitas justamente por conta de algum erro que possa ocorrer próximo da finalização da execução completa.

De forma particular em nosso laboratório, tivemos experiências desagradáveis quando após algumas semanas de processamento uma das estações de trabalho, por algum motivo, parava de funcionar. Isto nos forçava a executar tudo novamente, desconsiderando todo o processamento feito antes da falha. O mesmo ocorria quando o sistema parava por falta de energia, ou a máquina de gerência parava o funcionamento por alguma falha no sistema operacional. Embora ocorrido conosco, acreditamos este ser também um problema de outros laboratórios.

São inúmeros os problemas que podem ocorrer em um ambiente paralelo durante a execução do BLAST (5), variando desde problemas de *hardware* até problemas gerados por programas externos em processamento concorrente.

2.5 Conclusão

Neste Capítulo apresentamos as principais requisitos tratados no decorrer desta dissertação referente ao processamento paralelo do BLAST. Assim sendo, iniciamos comentando os diferentes tempos de execução da ferramenta BLAST, e que uma forma de obter ganhos em execuções para o alto número de seqüências de consulta foi a utilização de um agrupamento de computadores. Foi discutida também a importância em se manter o tempo de execução balanceado entre as máquinas de trabalho, a fim de obter ganho efetivo no processamento como um todo.

A idéia do BLAST em ambiente paralelo ou distribuído se tornou amadurecida até o ponto em que fragmentos distintos da base de dados fossem tratados em separado por cada estação de trabalho, reduzindo-se o acesso ao disco e paralelizando-o. Contudo, ainda assim o uso da estratégia fragmentada

implica em custos adicionais em relação à estratégia replicada.

Neste Capítulo distinguimos algumas dificuldades e/ou carências importantes para estratégias de execução da ferramenta BLAST, considerando principalmente alocação fragmentada da base de dados e alguns casos gerais pertinentes à alocação replicada, são elas:

- Necessidade em manter o balanceamento de carga entre as máquinas, não só considerando fatores externos, mas também os intrínsecos ao processamento BLAST;
- Explorar o uso de granularidades no processamento das seqüências de consulta e da base de dados, a fim de que o balanceamento seja o melhor possível;
- Aperfeiçoar o acesso ao disco, a fim de suportar grandes bancos de dados;
- Evitar custos adicionais de pré-processamento em estratégias fragmentadas;
- Possibilitar precisão das estatísticas de alinhamento, mesmo que em bases fragmentadas;
- Manter o processamento da ferramenta ao mesmo tempo em que a base de dados é alterada;
- Permitir que a execução seja robusta, mesmo em caso de falhas.

No Capítulo 4 vários trabalhos relacionados são citados, e que de alguma forma abordam algumas dessas dificuldades na execução paralela do BLAST relatas até o momento.