FastTree2.1 Parallelization

Morgan Price, Dehal Paramvir and Adam Arkin

December 4, 2018

Overview

1 Background

2 Modifying the Source

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining
- Uses tree rearrangements at each step in order to increase likelihood

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining
- Uses tree rearrangements at each step in order to increase likelihood
- FAST!

Table 4. Running time and memory usage on genuine alignments.

	Distinct		FastTree 2.0.0			RAxML 7	PhyML 3
Alignment	**Sequences**	**Positions**	**Model**	**Hours**	**GB**	**Hours**	**Hours**
16S rRNA, subsets	500	1,287 nt.	GTR	0.02	–	2.2	2.9
COGs, subsets	500	65–1,009 a.a.	JTT	0.02	–	5.2	7.2
COGs, subsets	2,500	197–384 a.a.	JTT	0.11	–	61	–
Efflux permeases	8,362	394 a.a.	JTT	0.25	0.35	197	>1,200
16S rRNAs, families	15,011	1,287 nt.	GTR	0.66	0.56	64	>2,000
ABC transporters	39,092	214 a.a.	JTT	1.02	0.96	–	–
16S rRNAs, all	237,882	1,287 nt.	JC	21.8	5.8	–	–

All runs used a single thread of execution. All runs accounted for variable rates across sites, using CAT for RAxML 7 and FastTree 2 or Γ_4 for PhyML 3. All FastTree runs include local SH-like supports and all RAxML runs include branch lengths under Γ_4. RAxML and PhyML were run without support values (no bootstrap). For random subsets of 500 16S rRNAs or for COGs, we show average running times. For alignments with over 1,000 sequences, we used RAxML 7.2.1's fast convergence option.
doi:10.1371/journal.pone.0009490.t004

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining
- Uses tree rearrangements at each step in order to increase likelihood

What is FastTree2.1?

- A maximum likelihood method for inferring phylogenies via sequence alignments
- Starting tree obtained via heuristic neighbor joining
- Uses tree rearrangements at each step in order to increase likelihood
- FAST!

Table 4. Running time and memory usage on genuine alignments.

	Distinct		FastTree 2.0.0			RAxML 7	PhyML 3
Alignment	**Sequences**	**Positions**	**Model**	**Hours**	**GB**	**Hours**	**Hours**
16S rRNA, subsets	500	1,287 nt.	GTR	0.02	–	2.2	2.9
COGs, subsets	500	65–1,009 a.a.	JTT	0.02	–	5.2	7.2
COGs, subsets	2,500	197–384 a.a.	JTT	0.11	–	61	–
Efflux permeases	8,362	394 a.a.	JTT	0.25	0.35	197	>1,200
16S rRNAs, families	15,011	1,287 nt.	GTR	0.66	0.56	64	>2,000
ABC transporters	39,092	214 a.a.	JTT	1.02	0.96	–	–
16S rRNAs, all	237,882	1,287 nt.	JC	21.8	5.8	–	–

All runs used a single thread of execution. All runs accounted for variable rates across sites, using CAT for RAxML 7 and FastTree 2 or Γ_4 for PhyML 3. All FastTree runs include local SH-like supports and all RAxML runs include branch lengths under Γ_4. RAxML and PhyML were run without support values (no bootstrap). For random subsets of 500 16S rRNAs or for COGs, we show average running times. For alignments with over 1,000 sequences, we used RAxML 7.2.1's fast convergence option.
doi:10.1371/journal.pone.0009490.t004

Why is FastTree so fast?

From the FastTree website:

- FastTree considers only NNIs, not SPR moves

Why is FastTree so fast?

From the FastTree website:

- FastTree considers only NNIs, not SPR moves
- FastTree maintains only one topology at a time

Why is FastTree so fast?

From the FastTree website:

- FastTree considers only NNIs, not SPR moves
- FastTree maintains only one topology at a time
- FastTree only optimizes each individual branch length to an accuracy of 0.0001 or 0.1% of the branch length, whichever is greater

Why is FastTree so fast?

From the FastTree website:

- FastTree considers only NNIs, not SPR moves
- FastTree maintains only one topology at a time
- FastTree only optimizes each individual branch length to an accuracy of 0.0001 or 0.1% of the branch length, whichever is greater
- FastTree limits the number of rounds of NNIs (default: $2 * \log_2(N)$ rounds) to ensure a predictable running time. However, FastTree normally converges before reaching this limit.

Why doesn't everyone use FastTree?

Figure: Ran on a single core.

Table 1. Topological accuracy of trees inferred from simulated alignments.

	250	1,250	5,000	78,132
Method	**a.a.**	**a.a.**	**a.a.**	**nt.**
RAxML 7 (JTT+CAT, SPRs)	90.5%	88.4%	88.4%	–
PhyML 3.0 (JTT+Γ_4, SPRs)	89.9%	–	–	–
FastTree 2.0.0 (JTT+CAT or JC+CAT)	86.9%	83.7%	84.3%	92.1%
PhyML 3.0 (JTT+Γ_4, no SPRs)	86.0%	–	–	–

- RAxML and PhyML are more accurate than FastTree

Why doesn't everyone use FastTree?

Figure: Ran on a single core.

Table 1. Topological accuracy of trees inferred from simulated alignments.

	250	1,250	5,000	78,132
Method	**a.a.**	**a.a.**	**a.a.**	**nt.**
RAxML 7 (JTT+CAT, SPRs)	90.5%	88.4%	88.4%	–
PhyML 3.0 (JTT+Γ_4, SPRs)	89.9%	–	–	–
FastTree 2.0.0 (JTT+CAT or JC+CAT)	86.9%	83.7%	84.3%	92.1%
PhyML 3.0 (JTT+Γ_4, no SPRs)	86.0%	–	–	–

- RAxML and PhyML are more accurate than FastTree
- RAxML supports both HPC and shared-memory parallelism

Why doesn't everyone use FastTree?

Figure: Ran on a single core.

Table 1. Topological accuracy of trees inferred from simulated alignments.

	250	1,250	5,000	78,132
Method	**a.a.**	**a.a.**	**a.a.**	**nt.**
RAxML 7 (JTT+CAT, SPRs)	90.5%	88.4%	88.4%	–
PhyML 3.0 (JTT+Γ_4, SPRs)	89.9%	–	–	–
FastTree 2.0.0 (JTT+CAT or JC+CAT)	86.9%	83.7%	84.3%	92.1%
PhyML 3.0 (JTT+Γ_4, no SPRs)	86.0%	–	–	–

- RAxML and PhyML are more accurate than FastTree
- RAxML supports both HPC and shared-memory parallelism
- PhyML supports HPC for bootstrap computation

Why doesn't everyone use FastTree?

Figure: Ran on a single core.

Table 1. Topological accuracy of trees inferred from simulated alignments.

	250	1,250	5,000	78,132
Method	**a.a.**	**a.a.**	**a.a.**	**nt.**
RAxML 7 (JTT+CAT, SPRs)	90.5%	88.4%	88.4%	–
PhyML 3.0 (JTT+Γ_4, SPRs)	89.9%	–	–	–
FastTree 2.0.0 (JTT+CAT or JC+CAT)	86.9%	83.7%	84.3%	92.1%
PhyML 3.0 (JTT+Γ_4, no SPRs)	86.0%	–	–	–

- RAxML and PhyML are more accurate than FastTree
- RAxML supports both HPC and shared-memory parallelism
- PhyML supports HPC for bootstrap computation
- FastTree only can scale to at most 3 cores during the ML stage

Observations

- The source for FastTree2.1 is 10,000 lines

Observations

- The source for FastTree2.1 is 10,000 lines
- 7% is comments

Observations

- The source for FastTree2.1 is 10,000 lines
- 7% is comments
- All in one file

Observations

- The source for FastTree2.1 is 10,000 lines
- 7% is comments
- All in one file
- Numerical overflow in computation can be finicky

Goals

- Refactor into multiple files

Goals

- Refactor into multiple files
- Create makefile for building

Goals

- Refactor into multiple files
- Create makefile for building
- Parallelize likelihood computation in order to replace the course grained parallelism that limits core usage to 3 during ML computation.

Parallelizing FastTree

There were two methods proposed to parallelize the likelihood computation of FastTree:

1. Morgan Price suggested that for every round of NNIs, the tree be split into n subtrees (for the n cores) and have each subtree be individually optimized, and then stitched back together.

Parallelizing FastTree

There were two methods proposed to parallelize the likelihood computation of FastTree:

1. Morgan Price suggested that for every round of NNIs, the tree be split into n subtrees (for the n cores) and have each subtree be individually optimized, and then stitched back together.
2. Erin Molloy from the Warnow Lab suggested that a scheme of parallelization similar to that of RAxML be attempted, which would be a fine-grain approach that parallelizes the computation of the likelihood of each site.

Current Stage and Predictions

- Part of the likelihood computation has been parallelized (for the Jukes-Cantor model). However, the remaining two models (nucleotide and aa transition matrices) need a bit more attention, as a core data structure relies on sequential computation.
- A simple way to attempt to break the sequential dependency of the structure would be to do an initial scan and create a mapping of the resources for each site, however this itself would be timely and sequential. We may use a parallel prefix to determine the mapping in order to leverage multiple cores.
- An interesting benchmark is to see how long sequences need to be to benefit from a certain degree of parallelism. The one advantage of the tree-splitting parallel schema is that it doesn't rely on sequence length for effectiveness.

Questions?