STAR Index - Comprehensive Reference Guide

Overview

STAR (Spliced Transcripts Alignment to a Reference) is an ultrafast RNA-seq aligner that performs splice-aware alignment of reads to a reference genome. The STAR index is a pre-computed data structure that enables this rapid alignment.

Website: https://github.com/alexdobin/STAR
Publication: Dobin et al. Bioinformatics 2013

Applicable to: RNA-seq, small RNA-seq, long-read RNA-seq, fusion detection

---

What is a STAR Index?

The STAR index is a collection of pre-computed data structures that enable fast sequence alignment:

Core Components

1. **Suffix Array (SA)** - Core data structure mapping genome positions
2. **SAindex** - Index into the suffix array for quick lookups
3. **Genome** - Packed binary representation of genome sequence
4. **Splice Junction Database (sjdb)** - Annotated splice junctions from GTF
5. **Chromosome metadata** - Names, lengths, positions

Why Pre-indexing?

Without index:

- Every alignment requires scanning entire genome
- Prohibitively slow for large genomes
- ~Hours per sample

With index:

- Index loaded into RAM once
- All lookups are instant hash/array operations
- ~10-30 minutes per sample

STAR vs Other Aligners

Feature	STAR	HISAT2	TopHat2 (deprecated)
Speed	Very fast	Fast	Slow

Feature	STAR	HISAT2	TopHat2 (deprecated)
Memory	High (30GB)	Moderate (8GB)	Moderate
Splice detection	Excellent	Excellent	Good
Index size	Large (30GB)	Moderate (8GB)	Moderate
Novel junction	Two-pass mode	Yes	Yes
Long reads	Excellent	Good	Poor
Accuracy	Excellent	Excellent	Good

When to use STAR:

- Standard RNA-seq workflows
- High accuracy needed
- Sufficient memory available (30GB+)
- Splice junction analysis
- Fusion detection

When to use alternatives:

- Limited memory (<16GB) → HISAT2
- DNA-seq alignment → BWA, Bowtie2
- Ultra-long reads (>10kb) → Minimap2

Building a STAR Index

Basic Command

```
STAR --runMode genomeGenerate \
    --runThreadN 8 \
    --genomeDir star_index \
    --genomeFastaFiles genome.fa \
    --sjdbGTFfile annotation.gtf \
    --sjdbOverhang 100 \
    --genomeSAindexNbases 14
```

Command Breakdown

```
STAR \                                  # STAR aligner program
    --runMode genomeGenerate \          # Mode: generate index (not align)
    --runThreadN 8 \                    # Use 8 CPU cores for parallel indexing
    --genomeDir star_index \            # Output directory (will be created)
    --genomeFastaFiles genome.fa \      # Input genome FASTA file(s)
    --sjdbGTFfile annotation.gtf \      # Gene annotation for splice junctions
    --sjdbOverhang 100 \                # Max overhang for junctions (ReadLength-1)
    --genomeSAindexNbases 14            # Suffix array sparsity parameter
```

What happens during indexing:

1. **Genome loading**: STAR reads genome FASTA file(s)
2. **Suffix array construction**: Builds SA data structure (slowest step)
3. **GTF parsing**: Extracts exon coordinates and splice junctions
4. **Junction database**: Creates sjdb with flanking sequences
5. **Index writing**: Saves all structures to genomeDir/
6. **Validation**: Checks index integrity

Typical runtime:

- Human genome (3Gb): 1-2 hours on 8 cores
- Mouse genome (2.5Gb): 45-90 minutes on 8 cores
- Drosophila (140Mb): 5-10 minutes on 8 cores
- Yeast (12Mb): 1-2 minutes on 8 cores

Critical Parameters

--sjdbOverhang (Splice Junction Overhang)

What it is: Maximum overhang on each side of a splice junction

How to set: `ReadLength - 1`

Examples:

```
50bp reads  → --sjdbOverhang 49
75bp reads  → --sjdbOverhang 74
100bp reads → --sjdbOverhang 99
150bp reads → --sjdbOverhang 149
```

Why it matters:

- Affects splice junction detection sensitivity
- Too short: Miss junctions near read ends
- Too long: No benefit, just wastes memory
- Value of 100 works well for 75-150bp reads (common compromise)

Default: 100 (if not specified)

Multi-length reads: If you have mixed read lengths (e.g., 75bp and 100bp):

- Use: `max(ReadLength) - 1`
- Or: 100 (universal compromise)

--genomeSAindexNbases (Suffix Array Sparsity)

What it is: Controls spacing of suffix array entries (affects memory/speed)

Formula: `min(14, log2(GenomeSize)/2 - 1)`

Calculation examples:

```
# Human genome (3 billion bp)
log2(3,000,000,000) / 2 - 1 = 31.5 / 2 - 1 = 14.75 → 14

# Mouse genome (2.5 billion bp)
log2(2,500,000,000) / 2 - 1 = 31.2 / 2 - 1 = 14.6 → 14

# Drosophila (140 million bp)
log2(140,000,000) / 2 - 1 = 27.1 / 2 - 1 = 12.5 → 12

# C. elegans (100 million bp)
log2(100,000,000) / 2 - 1 = 26.6 / 2 - 1 = 12.3 → 12

# Yeast (12 million bp)
log2(12,000,000) / 2 - 1 = 23.5 / 2 - 1 = 10.7 → 10
```

Effects:

Value	Memory Usage	Speed	Genome Size
14	Highest	Fastest	Human/Mouse (>2Gb)
13	High	Fast	1-2Gb
12	Moderate	Moderate	100Mb-1Gb
11	Low	Slow	50-100Mb
10	Very low	Slower	<50Mb

When to adjust:

- **Reduce** if "SA size error" occurs (genome too small)
- **Reduce** if running out of memory during indexing
- **Never increase** beyond 14 (no benefit)

Default: Auto-calculated by STAR (usually correct)

Index Output Structure

After indexing, `star_index/` contains:

```
star_index/
   Genome                    # Packed binary genome sequence (~3GB for human)
   SA                        # Suffix array (LARGEST: ~20-25GB for human)
   SAindex                   # Suffix array index (~1.5GB)
   chrName.txt               # Chromosome names (human-readable)
   chrNameLength.txt         # Chromosome name lengths
   chrLength.txt             # Chromosome lengths in bases
   chrStart.txt              # Starting byte positions in Genome file
   chrName.txt               # List of chromosomes
```

```
genomeParameters.txt        # Index build parameters
sjdbInfo.txt                # Splice junction database info
sjdbList.fromGTF.out.tab    # Junctions extracted from GTF
sjdbList.out.tab            # Final junction list (used for alignment)
exonGeTrInfo.tab            # Exon/gene/transcript relationships
exonInfo.tab                # Exon coordinates
geneInfo.tab                # Gene information
transcriptInfo.tab          # Transcript information
Log.out                     # Indexing log file
```

Important Files Explained

SA (Suffix Array):

- Largest file (~20-25GB for human)
- Core data structure for fast sequence search
- Maps every genome position for quick lookup
- Loaded into RAM during alignment

Genome:

- Packed binary genome sequence
- More compact than FASTA
- Loaded into RAM during alignment

sjdbList.out.tab:

- Splice junctions from GTF
- Format: `chr start end strand motif annotated`
- Used during alignment to map spliced reads

genomeParameters.txt (human-readable):

```
versionGenome   2.7.10a
genomeFastaFiles    genome.fa
genomeSAindexNbases 14
genomeChrBinNbits   18
genomeSAsparseD 1
sjdbOverhang    100
sjdbFileChrStartEnd -
sjdbGTFfile annotation.gtf
sjdbGTFchrPrefix    -
sjdbGTFfeatureExon  exon
sjdbGTFtagExonParentTranscript  transcript_id
sjdbGTFtagExonParentGene    gene_id
```

Useful for:

- Verifying index parameters
- Checking STAR version compatibility

- Troubleshooting indexing issues

Resource Requirements

Memory (RAM)

Genome	Index Build	During Alignment
Human	30-50GB	30GB (index in RAM)
Mouse	25-40GB	25GB
Rat	25-40GB	25GB
Drosophila	5-8GB	3GB
C. elegans	3-5GB	2GB
Yeast	2-3GB	1GB

Why so much memory?:

- STAR loads entire index into RAM
- Enables ultra-fast alignment (no disk I/O)
- Trade-off: Speed vs memory

If memory limited:

- Use HISAT2 instead (8GB for human)
- Build on machine with more RAM, then transfer index
- Use `--genomeSAsparseD 2` (slower alignment, less memory)

CPU Cores

Index building:

- Scales well up to 8-16 cores
- Diminishing returns beyond 16 cores
- Bottleneck shifts to disk I/O

Recommended:

- 8 cores: Good balance
- 16 cores: Faster for large genomes
- 4 cores: Acceptable, ~2x slower

Disk Space

Temporary space during indexing:

- 2-3× genome size
- Human: 6-10GB temporary

Final index size:

Genome	Index Size
Human	25-30GB
Mouse	20-25GB
Rat	20-25GB
Drosophila	2-3GB
C. elegans	1-2GB
Yeast	200-400MB

Storage considerations:

- Use fast local storage (SSD) during indexing
- Can be on slower storage for long-term (index loaded to RAM anyway)
- Network storage acceptable for published index

Time

Genome	8 cores	16 cores
Human	1-2 hours	45-90 min
Mouse	45-90 min	30-60 min
Drosophila	5-10 min	3-5 min
Yeast	1-2 min	<1 min

Factors affecting speed:

- Disk I/O speed (SSD vs HDD)
- CPU speed
- Number of cores
- Genome complexity (repeat content)

Advanced Options

Multiple Genome Files

If genome is split across files:

```
STAR --runMode genomeGenerate \
    --genomeFastaFiles chr1.fa chr2.fa chr3.fa \
```

```
    # OR
    --genomeFastaFiles genome_part*.fa \
    ...
```

Use case: Some databases provide chromosomes as separate files

---

Custom Chromosome Names

Add prefix to chromosome names:

```
STAR --runMode genomeGenerate \
    --genomeFastaFiles genome.fa \
    --sjdbGTFfile annotation.gtf \
    --sjdbGTFchrPrefix "chr" \
    ...
```

Use case:

- GTF has "chr1", FASTA has "1"
- Adds "chr" prefix to GTF chromosomes
- Alternative: Fix files beforehand

Sparse Suffix Array

Reduce memory during alignment (slower):

```
STAR --runMode genomeGenerate \
    --genomeFastaFiles genome.fa \
    --genomeSAsparseD 2 \
    ...
```

Effect:

- SA sampled every 2 bases (vs every base)
- Reduces memory ~50%
- Alignment ~20% slower

Use case: Memory-constrained systems

Without GTF (Alignment-Only Mode)

Build index without splice junctions:

```
STAR --runMode genomeGenerate \
    --genomeFastaFiles genome.fa \
    --genomeDir star_index
```

When to use:

- No annotation available
- Genomic DNA alignment (not RNA-seq)
- Will detect unannotated junctions during alignment

Trade-off:

- Can still detect junctions
- Less sensitive without annotation
- Novel junction discovery still works

Index Compatibility and Reusability

When to Reuse Index

Can reuse if:

- Same genome assembly version
- Same gene annotation (or minor update)
- Same STAR version (or compatible)
- Same or similar read lengths

Must rebuild if:

- Different genome assembly (GRCh37 → GRCh38)
- Major annotation update (e.g., GENCODE v19 → v38)
- STAR major version change
- Very different read lengths (75bp → 300bp)

STAR Version Compatibility

Generally compatible:

- Within same major version (2.7.x → 2.7.y)
- Minor version bumps usually OK

May need rebuild:

- Major version change (2.6.x → 2.7.x)
- Check STAR release notes

Check index STAR version:

```
grep "versionGenome" star_index/genomeParameters.txt
```

Cross-Platform Compatibility

Compatible between:

- Different Linux distributions
- Linux → macOS (usually)
- Different filesystems

NOT compatible:

- Different endianness (rare)
- 32-bit → 64-bit (or vice versa)

Troubleshooting

"EXITING because of FATAL ERROR: Genome version is incompatible"

Cause: Index built with different STAR version

Solution:

```
# Check index STAR version
cat star_index/genomeParameters.txt | grep versionGenome

# Check current STAR version
STAR --version

# Rebuild if versions incompatible
```

"EXITING because of FATAL ERROR: genome size error"

Cause: `genomeSAindexNbases` too large for small genome

Solution:

```
# Calculate correct value
genome_size=$(grep -v ">" genome.fa | tr -d '\n' | wc -c)
nbases=$(echo "l($genome_size)/l(2)/2 - 1" | bc -l | cut -d'.' -f1)

# Rebuild with correct value
STAR --runMode genomeGenerate \
    --genomeSAindexNbases $nbases \
    ...
```

Or manually reduce:

- Try 13 (if was 14)

- Try 12 (if still fails)
- Continue reducing until success

"Out of memory" During Indexing

Cause: Insufficient RAM

Solutions:

1. **Close other applications**

   ```
   # Check memory usage
   free -h
   top
   ```

2. **Reduce genomeSAindexNbases**

   ```
   # Try one less
   STAR --runMode genomeGenerate --genomeSAindexNbases 13 ...
   ```

3. **Use sparse indexing**

   ```
   STAR --runMode genomeGenerate --genomeSAsparseD 2 ...
   ```

4. **Use machine with more RAM**

   - Build on HPC node
   - Use cloud instance with more memory
   - Transfer index to local machine

GTF Parsing Errors

Error: "WARNING: --sjdbGTFfile: skipped lines due to errors"

Common causes:

1. **GFF3 instead of GTF**

   ```
   # Check format
   head -n 5 annotation.gtf

   # GTF has specific attribute format:
   # gene_id "ENSG..."; transcript_id "ENST...";

   # Convert GFF3 to GTF if needed
   gffread annotation.gff3 -T -o annotation.gtf
   ```

2. **Chromosome name mismatch**

```
# Check FASTA chromosomes
grep "^>" genome.fa | cut -d' ' -f1

# Check GTF chromosomes
cut -f1 annotation.gtf | sort -u

# Must match exactly!
```

3. **Compressed GTF**

```
# STAR cannot read .gz directly
gunzip annotation.gtf.gz
```

4. **Comment lines**

```
# Remove comment lines
grep -v "^#" annotation.gtf > annotation_clean.gtf
```

Very Slow Indexing (>3 hours)

Causes and solutions:

1. **Slow disk I/O**

```
# Check if using HDD (slow) vs SSD (fast)
# Build on local SSD, not network storage
```

2. **Not enough CPU cores**

```
# Increase threads
STAR --runMode genomeGenerate --runThreadN 16 ...
```

3. **Swapping to disk** (not enough RAM)

```
# Check swap usage
free -h
vmstat 1

# If swapping, need more RAM or reduce genomeSAindexNbases
```

Best Practices

For Production Use

Always:

- Include GTF annotation (more accurate)
- Use appropriate sjdbOverhang for read length
- Keep index build logs

- Document STAR version
- Test with small dataset first

Recommend:

- Build once, publish to shared location
- Use descriptive index directory names (genome_version_annotation)
- Store genomeParameters.txt separately for reference
- Validate index after building (test alignment)

Avoid:

- Building on network storage (slow)
- Deleting build logs
- Using very old STAR versions
- Guessing genomeSAindexNbases (let STAR calculate)

Naming Conventions

Good examples:

```
GRCh38_gencode_v38/
GRCm39_ensembl_110/
hg19_ucsc_refseq/
```

Include:

- Genome assembly version
- Annotation source
- Annotation version

Why:

- Prevents confusion
- Easy to identify correct index
- Facilitates sharing

Index Organization

Recommended structure:

```
reference_genomes/
  human/
      GRCh38/
          genome.fa
          annotation.gtf
          star_index/
      GRCh37/
```

```
        ...
  mouse/
      GRCm39/
          ...
  drosophila/
      dm6/
          ...
```

Command Reference

```bash
# Basic index build
STAR --runMode genomeGenerate \
    --genomeDir star_index \
    --genomeFastaFiles genome.fa \
    --sjdbGTFfile annotation.gtf

# With custom parameters
STAR --runMode genomeGenerate \
    --genomeDir star_index \
    --genomeFastaFiles genome.fa \
    --sjdbGTFfile annotation.gtf \
    --sjdbOverhang 149 \
    --genomeSAindexNbases 14 \
    --runThreadN 16

# Sparse index (low memory)
STAR --runMode genomeGenerate \
    --genomeDir star_index \
    --genomeFastaFiles genome.fa \
    --sjdbGTFfile annotation.gtf \
    --genomeSAsparseD 2

# Without annotation
STAR --runMode genomeGenerate \
    --genomeDir star_index \
    --genomeFastaFiles genome.fa

# Check index version
cat star_index/genomeParameters.txt | grep versionGenome

# Check index size
du -sh star_index/

# Validate index files exist
```

```
ls -lh star_index/SA
ls -lh star_index/Genome
```

Related Documentation

- **STAR Alignment**: `docs/star_align.md`
- **Gene Counting**: `docs/feature_counts.md`
- **Splice Junction Analysis**: `docs/splice_analysis.md`
- **STAR Manual**: https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf

---

Document Version: 2.0
Last Updated: January 2026
STAR Version: 2.7.10a+
Applicable to: All RNA-seq applications requiring genome alignment